Optimal Steel Temperature Control on the Run-Out Table

From single setup to sample control

E. Verboom

TU Delft

Optimal Steel Temperature Control on the Run-Out Table

From single setup to sample control

by

E. Verboom

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday January 13, 2021 at 2:00 PM.

Student number:	4315488	
Project duration:	May 1, 2020 – January 13, 2021	
Thesis committee:	Dr. N. Yorke-Smith,	TU Delft (supervisor)
	Prof. Dr. P. A. N. Bosman,	TU Delft
	Ir. M. Bsibsi	Tata Steel

Preface

Before you lies the thesis *Optimal steel temperature control on the Run-Out Table, from single setup to sample control*, a report on the work done as graduate intern at Tata Steel. It has been written to meet the graduation requirements for the master Computer Science at the Delft University of Technology. I have been working on this project from May 2020 until the start of January 2021. The project has been a great combination of a physical background and an algorithmic approach. Further, the clear application of the results kept me driven.

The year 2020 has been a year none of us will soon forget and I can only say I had the privilege to have been at the manufacturing site of Tata Steel IJmuiden once, for my interview before starting the project. Still full of hope that it would just be a few weeks working from home before returning and getting to know the company after my literature research. You can guess nothing of the sort happened and I ended up doing the entire project from my own apartment. The job interview with Mustapha Bsibsi, Knowledge Group Leader 'Thermal Processes', was initially for another project, which did not quite live up to my expectations. With a lot of flexibility and willingness, Mustapha presented an optimization problem in the finishing mill where steel slabs are cooled. The physicist in me immediately awoke and we were both excited to start. The result of this excitement, this thesis, would have never been what it has become without the help of some people, whom I would like to give credit for that.

First of all, I would like to thank my supervisor Neil Yorke-Smith, from the TU Delft. He has always been supportive and kept asking me the right questions to keep going. Next, I would like to thank Peter Bosman in advance for being the second academic member of my thesis committee and for taking the time to read and judge my work. Also, I would like to thank Tata Steel for giving me the opportunity to do a graduate internship in these unstable times. A special thanks to Mustapha Bsibsi for his guidance and to Ramon Speets for helping me to understand the systems and methods used at the Run-Out Table from a distance. Especially since working from home, with kids full of energy and needs, has not made it easier for any of you.

Neither has working from home been easy on me. With not getting the experience of working on a project within a company being the least concern of all. As a team player I missed the daily sparring sessions with friends or colleagues and would have loved to thank them for the many coffee breaks here. Instead I would like to thank Thijs Greevink for having lunch with me every day, listening to all my struggles and helping me to put everything in perspective from time to time.

I hope you enjoy your reading.

E. Verboom
Delft, December 2020

Abstract

This thesis investigates the optimization algorithm used for steel temperature control on the run-out table at Tata Steel IJmuiden. The system currently implemented in the finishing mill is called STORM (Smart Temperature Optimization on the Run out table for Mechanical property control) and was created and fully implemented this year. This control system computes one setup based on the estimated maximum speed and properties of the head of the slab at the time of entering the finishing mill and is based on a Genetic Algorithm (GA). To accommodate for variations over the length of a steel slab, manual action is required. In order to automate this process, the aim is to further develop the system to enable computation of setups for multiple samples over the length of the slab within the limited time available between a slab entering the finishing mill and the start of the cooling process.

For this purpose, this thesis introduces three precomputation extensions that incorporate the concept of a warm start with different methods to incorporate knowledge of the problem: STORM-Trained on a Single Model, STORM-Trained on Multiple Models and STORM-Trained on Multiple Objectives. The baseline is taken to be STORM-Uninformed, that uses no prior knowledge to compute multiple setups. The new methods are compared based on four different product recipes with respect to the fitness of the optimal setup found by the solver and the number of iterations required to obtain a useful setup. A setup is assumed to be useful when it is at most 1.2 times the known optimum of a problem instance.

The experiments conducted to compare the three mentioned extensions show that when optimizing for samples over the length of the run-out table STORM-TMM outperforms both other methods, though for problem instances with two domains a population size of 200 is required for precomputation and even then not always a useful setup is found within the iteration limit. For optimization for different sets of objectives there is an even greater distinction in the results for problem instances with one and two domains. For the former, the multi-objective algorithm STORM-TMO most definitely performs best, where for the latter STORM-TSM performs best. Again when computing a setup for two domains, a useful setup is not always found. In conclusion, the multi-objective methods STORM-TMM and STORM-TMO can be used for sample setup computation for problems with one domain after some final checks. For problems with multiple domains, additional research is required.

Contents

Acronyms

CR	Cooling Rate
CRT	Cooling Rate Target
CT	Coiling Temperature
CTT	Coiling Temperature Target
GA	Genetic Algorithm
IT	Intermediate Temperature
ITT	Intermediate Temperature Target
MOGA	Multi-objective Genetic Algorithm
NSGA-II	Non dominated Search Genetic Algorithm II
ROT	Run-out Table
STORM	Smart Temperature Optimization on the Run-out table for Mechanical property control
STORM-TMM	STORM Trained on Multiple Models
STORM-TMO	STORM Trained on Multiple Objectives
STORM-TSM	STORM Trained on a Single Model
STORM-U	STORM Uninformed

Glossary

Active cooling zone	The group of sections between and together with the first and last section that are turned on within a given domain
Black box	General description of an object of which the behaviour is known, but the inner mechanism is not. In this context the process assigning a fitness value to a Setup is considered to be a black box, because there is no simple relation between the two
Child	New Individual created from two Parents
Coiling temperature	Temperature of the steel at the end of the cooling process, where it is coiled
Control sequence	Sequence containing the control values for the run-out table, to say which water banks are used at what setting
Convergence	A solver is said to converge when it has reached the most optimal value of the objective function it is able to find
Convexity	Convexity is a property of a mathematical relationship or function. When a function is convex, a line drawn between any two points on the graph of the function lies above the graph
Cooling path	The course of the temperature of a steel slab over time, during the cooling process
Cooling pattern	Set of cooling banks included in a setup, see Setup
Cooling rate	Temperature change per second
Cooling section	The position of a set of cooling banks on top and bottom
Elite set	Set of promising Individuals
Fitness	Measure of the quality of an Individual
Flow rate	The water flow setting of a water bank, where 0 means no flow, 1 means flow at half capacity and 2 at full capacity. The capacity of the bank is known by the operator and can vary over the Run-out table
Gene	Single variable used to describe part of a solution of the defined problem
Gene sequence	Set of Genes that can be mapped to a solution of the defined problem

Genetic Algorithm	An algorithm that functions according to the process of genetic evolution
Individual	Single solution for a defined problem, encoded in a particular Gene sequence
Intermediate temperature	Temperature measured at a given section between the first and last section of the Run-out table
Linearity	Linearity is the property of a mathematical relationship or function that can be graphically represented as a straight line
Load	Cooling load for all considered section
Multi-objective Genetic Algorithm	A Genetic Algorithm that optimizes for different objectives in parallel, which outputs a population that tries to approach the Pareto front
Non dominated Search Genetic Algorithm II	A well known implementation of a Multi-objective Genetic Algorithm
Objective	A target for an optimization problem, expressed as a mathematical function
P-value	Probability a result is inflicted by chance
Parent	An Individual used to construct new individuals
Pareto front	Set of all Pareto efficient solutions. A solution is Pareto efficient when no other solution performs better all objectives
Phase fraction	Calculated phase fraction at the end of the section
Population	Set of Individuals
Predictive modelling	A technique that uses a model to predict the future state of a system
Recipe	Combination of temperature targets, specific for certain products
Reference value	Value used as an approximation of the optimal value for a particular problem instance, obtained by running performing random search for one hour
Reference value	Value used as a baseline, to represent the optimal value of a particular problem instance. It is possible that the actual optimal value is lower, but there is no way to exactly determine the lowest possible value
Run-out table	Large installation of rollers over which the steel is transported, while it is cooled with water. This installation is located after the hot-rolling mill
Sample	Segment of a steel slab with a predefined length
Setup	A cooling pattern that defines which water banks are allowed to be used by the controller during cooling

Standard deviation	Measure of the amount of variation or dispersion within a set of values
Statistical significance	A determination about the null hypothesis, which hypothesizes that the results are due to chance alone. A result is statistically significant if the P-value is sufficiently small
STORM Trained Multiple Objectives	Implementation of STORM extended with a pre-computation based on a large set of contradictory objectives
STORM Trained on a Single Model	Implementation of STORM extended with a pre-computation based on the output values of the physical model with one set of values for the estimated maximum speed and finishing temperature
STORM Trained on Multiple Models	Implementation of STORM extended with a pre-computation based on the output values of the physical model with multiple sets of values for the estimated maximum speed and finishing temperature
STORM Uninformed	Implementation of STORM that uses an uninformed population as a starting point. This algorithm is used as a baseline to compare the other implementations to.
Strategy	The combination of a set of objectives and a domain definition
Surface temperature	Calculated strip surface temperature at the end of the section
T-test	Statistical test to determine the statistical significance of the difference between means of two or more groups of data
Temperature	Refers to the calculated average strip temperature at the end of a section
Temperature point	Calculated average strip temperature at the end of a strategy domain
Transfer point	First section at which a certain criterion is met, mostly the temperature of the steel falling below a predefined value
Transformation fraction	See Phase fraction
Warm start	A solver has a warm, or informed, start when it is given additional information on where to find the best solution before computation starts

1 Introduction

Tata Steel IJmuiden is one of Europe's leading steel manufacturing plants. The company supplies, among others, high-quality strip steel products to demanding markets such as construction, automotive and packaging. In the production of these products, there are many processes that require thermal control and the precision of temperature control highly affects the quality of the steel. One of these control processes can be found in the hot strip mill. After thick steel slabs are hot rolled to long strips with the required thickness, ranging from 2 to 25 mm, the slabs have to be cooled from about 900°C to a target temperature between 150 and 700°C on the run-out table (ROT).

During the cooling process the steel obtains its final mechanical properties. Newly developed products, like dual-phase, bainitic and complex-phase steel types, therefore require a precise and highly flexible control of the cooling path on the ROT. Not only the final temperature, or coiling temperature, but also the cooling rates and intermediate temperatures are important in achieving the right properties in the end.

The ROT is equipped with a total of 124 banks, half of them above and half of them under the strip, having a total length of 130 metres. A cooling bank has several pipes from which water jets are formed that hit the hot steel. Each individual bank has a valve that is switched by a controller to regulate the amount of water hitting the steel strip and thereby the amount of cooling. To be able to reach the desired cooling path the temperature control system uses predictive modelling to compute the temperatures of the strip along the ROT, where the optimal control sequence is determined by the minimum of an objective function. The result of the optimization process is the cooling pattern having the closest match to the desired cooling path of the steel strip, which is called the setup. An example of a possible setup configuration is depicted in Figure 1.1. During the cooling process the rotational speed of the ROT is changing and therefore not all cooling banks selected in the setup will be used at all times.

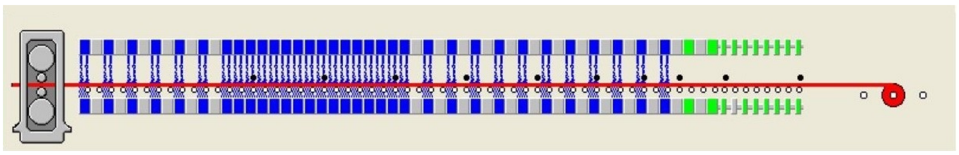


Figure 1.1: Example of a possible cooling pattern. Green: cooling bank is in the cooling pattern as defined in the setup phase. Blue: cooling bank is used to cool the strip, as determined by control. Grey: Cooling bank is not in the cooling pattern and is not used.

Before a steel strip enters the ROT, there is limited time available for the controller to determine the optimal setup. As there are many variables involved (e.g. the settings of each individual bank, material properties, velocity of the slab) of which some variables are discrete, it is very complex to find the minimum of the objective function. To accommodate for the advanced products currently processed in the hot strip mill, a new temperature control system called STORM (Smart Temperature Optimization on the Run out table for Mechanical property control) was created and fully implemented this year. This control system computes a setup based on the estimated maximum speed and properties of the head of the slab at the time of entering the finishing mill and is based on a Genetic Algorithm (GA). During cooling this setup functions as a starting point from which adjustments are made to accommodate for different speeds and feedback mechanisms come into play to adjust for minor variations in properties over the slab length. For all products currently produced this method works well. However the demand for more complex products requires constant innovation and asks for strategies that allow bigger variations within a steel slab. Therefore, the goal of this research is to find out if the current solution is able to compute multiple setups within the short time that is available between the moment a new slab enters the cooling installation and the moment cooling starts, or if it can be able to compute multiple setups within the time limit with some adjustment. This would allow for the computation of a single setup for each sample of the steel slab, to give leeway in sample variations and further automate the control process. In order to guide the search to such a solution, the following research questions are defined:

1. What characteristics influence the complexity of the problem?
2. Is the current method suitable to solve this particular problem and future applications?
3. Can the current method be extended to solve the problem more efficient?

The outline of this thesis is as follows, first chapter 2 describes how the problem of finding a suitable setup is defined and what the requirements for the solver are. Chapter 3 outlines the theory behind a Genetic Algorithm and the Multi-objective Genetic Algorithm used in this research. The methodology of the research performed is described in chapter 4 and introduces the three extensions of the current solution that have been examined: STORM-TSM, STORM-TMM and STORM-TMO. The performance of these implementations is compared to an uninformed GA and the last two are also compared to the first one. The results, statistical analysis and discussion can be found in chapter 5. Finally, chapter 6 summarizes the work done and poses suggestions for future steps towards implementation at Tata Steel.

2
Background

To answer the first research question as stated in chapter 1, it is important to understand what the problem is exactly and how it has to be solved. The aim of the solver is to find a cooling setup, i.e., which cooling banks to use at what intensity, such that the cooling path of the slab is as close to its optimal cooling path as possible. With the cooling path referring to the course of the steel's temperature over time. During the cooling process the ROT accelerates and decelerates, causing variance in the speed of the slab which has an impact on the required cooling setup. Also differences in inlet temperature and grain size impact the setup.

Currently computation of the setup is done by calculating the setup for the measured finishing temperature and the expected maximum speed of the ROT and linearly reducing the length of the pattern with respect to its speed. In order to find a solution for the highest speed, an optimizer is used to find the best solution within the given boundary that satisfies all constraints. The general variables describing a problem instance are described in section 2.1. Section 2.2 and section 2.3 go into detail on the domain and objectives respectively. The combination of a set of objectives and a domain is also called a *strategy*, which can be linked to particular products. Finally, section 2.4 summarizes the goal of the solver and the specific definition used to describe a solution.

2.1. Problem Variables

When a new slab enters the cooling system, measurement and prediction values are gathered to be used for computation of an optimal setup. These values can be divided into two categories: coil information, installation status and recipe. The coil information consists of properties of the steel slab entering the ROT, the installation status holds information about the ROT at the moment the slab is processed. The variables belonging to the components are the following:

- Coil information
  - Coil thickness, width and length
  - Expected finishing mill temperature
  - Expected maximum strip speed
  - Speed schedule
  - Chemical composition
  - Grain size
  - Adaptation parameters
- Installation status
  - Reference flow rates per section (at high and low setting)
  - Section operational status (available, unavailable or excluded)

2.2. Domain

The degrees of freedom for the setup are defined by the domain. A full description of the domain includes:

- The selection of cooling sections that may be used for the setup;
- The allowed flow rates: only high flow, only low flow or mixed flow;
- And symmetry settings.

More than one domain can be defined which allows two-step cooling and the definition of the trim section. The trim section consists of the last eight banks on top and bottom which are half the size of a main section and is often fixed to be in the pattern, such that these banks can be used for fine tuning. The properties describing a domain are summarized in Table 2.1.

Domain property	Symbol	Type	Description
DomainType		Integer	This identifies the type of target: main section (0) or trim section (1)
StartSection	$\mathrm{s_{min}}$	Float	The first section in the domain
EndSection	$\mathrm{s_{max}}$	Float	The last section in the domain
FlowLevelTop	$\mathrm{L_t}$	Integer	The flow level setting of the top banks, where 0 stands for mixed, 1 for low, 2 for high
FlowLevelBottom	$\mathrm{L_b}$	Integer	The flow level setting of the bottom banks, where 0 stands for mixed, 1 for low, 2 for high
Symmetry		Integer	The symmetry setting, see Table 2.2

Table 2.1: Domain properties, their symbol and description

SymmetryOption	Value	Description
ssoNone	0	No symmetry required
ssoFlow	1	Each switched top section also requires a switched bottom section with the same flow type
ssoSection	2	Each switched top section also requires a switched bottom section, but the flow level may differ

Table 2.2: Symmetry options and their integer values

2.3. Objective

The objective is a combination of desired targets related to temperature, cooling rate and transformation. Since some of the targets are defined by a *minimum* and/or a *maximum* value, the objective is somewhat soft. In STORM the total objective function is described by a weighted sum of partial objectives:

$$J(u) = w_1 \cdot J_1(u) + w_2 \cdot J_2(u) + \ldots + w_n \cdot J_n(u), \tag{2.1}$$

where u are the optimizer inputs (bank selection and flow-level), J_i is a partial objective function and w_i is the corresponding partial weighting. Each partial objective function J_i can be described according to the properties listed in Table 2.3.

The value of a partial objective function for given optimizer inputs can be visualised as the area under the curve in the corresponding section, that lies outside of the min-max value range. An example is depicted in Figure 2.1, where the green area contains all of the allowed values and the sum of the two red areas is equal to the value assigned to this objective. Practically, this means that the value of the objective function is a penalty, proportional to the deviation of the function value from the maximum or minimum desired value (whichever is closest). Mathematically this is defined as:

$$J_i(u) = w_i \cdot \frac{\sum_{j=s_{\min}}^{s_{\max}} \begin{cases} y(u)_j - r_{\max,i} & \text{for} \quad y(u)_j > r_{\max,i} \\ r_{\min,i} - y(u)_j & \text{for} \quad y(u)_j < r_{\min,i} \\ 0 & \text{for} \quad r_{\min,i} \leq y(u)_j \leq r_{\max,i} \end{cases}}{s_{\max,i} - s_{\min,i} + 1} \tag{2.2}$$

Objective property	Symbol	Description
Objective Type		This identifies the type of objective target, like average temperature target, cooling rate target, etc.
Minimum Value	r_{min}	The minimum desired value for this target or deviation from recipe value when applicable
Maximum Value	r_{max}	The maximum desired value for this target or deviation from recipe value when applicable
Start Section	s_{min}	The first section where this target is considered
End Section	s_{max}	The last section where this target is considered
Weight	w	The partial objective weight
Recipe Value	R	Recipe value from strategy that defines target value, which can be Coiling Temperature Target (CTT), Intermediate Temperature Target (ITT), Cooling Rate Target (CRT) or none
Transfer		This identifies how a temperature point, as defined in subsection 2.3.2, is to be used in this partial objective: as start section, end section or none of those

Table 2.3: Objective properties, their symbols and description

The values $y(u)_j$ depend on the objective type and are computed based on physical models that have been carefully developed and tweaked by Tata Steel. Simulation time is approximately instant and is thus not taken into account in this research. The different objective types are listed in Table 2.4 and discussed in subsection 2.3.1 to subsection 2.3.4.

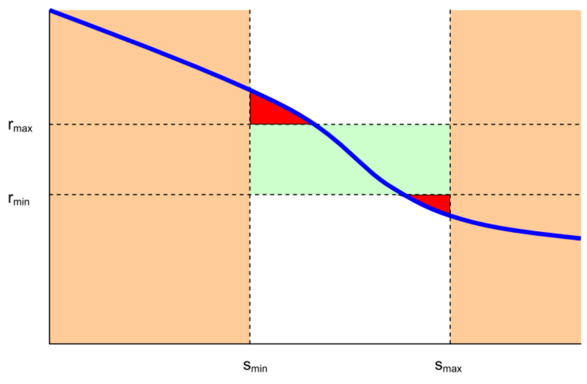


Figure 2.1: Visual representation of a penalty function

2.3.1. Temperature

The temperature objective defines the desired temperature for a specific cooling section (s). Generally, this objective is used to define the coiling temperature at the last cooling section. For this objective $y(u)$ equals the average strip temperature at the defined location depicted as $T_{mean,s}$.

2.3.2. Temperature Point

This objective type considers the position where a specific temperature boundary is crossed. The transfer point is stored, such that it can be used in other partial objectives and does not directly influence the value of the objective function. This temperature point can for example determine the boundary between two domains with different cooling rates.

2.3.3. Phase fractions

There are four phase fraction objectives, the fraction of steel that is transformed to ferrite, austenite, pearlite and bainite. All objective values are computed according to Equation 2.3 with $y(u)$ equal to the transformation fraction of the corresponding phase.

Objective type	Description
Temperature	The calculated average strip temperature at the end of the section
Temperature point	The calculated average strip temperature at the end of the strategy domain
Surface temperature	The calculated strip surface temperature at the end of the section
Load	The cooling load for all considered sections
Phase fractions	The calculated phase fraction at the end of the section
Cooling rate	The cooling rate of the active zone within the given domain

Table 2.4: Objective types and their description

2.3.4. Cooling Rate

In general the cooling rate is defined as the total temperature drop per second, i.e. °C/s, over the *active cooling zone*. This zone consists of a group of adjacent sections which are turned on within a given domain, see Figure 2.2 for two examples. Mathematically, the cooling rate is then defined as the temperature difference

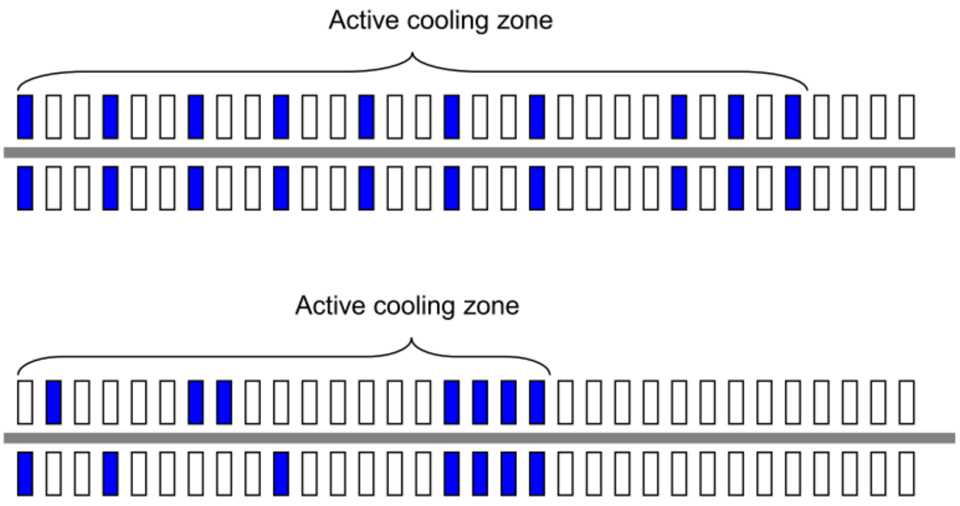


Figure 2.2: Examples of active cooling zones

over the active cooling zone divided by the residence time over the active cooling zone:

$$CR = \frac{\Delta T}{\Delta t}. \tag{2.3}$$

The temperature difference ΔT is determined as the difference between the entry and exit temperatures of the start and end sections of the active cooling zone. So this considers the temperature of a fixed sample point of the steel slab. Δt is the time it takes for this sample point to travel from the start section to the end section. Finally, the objective function value associated with the cooling rate results in

$$J(u) = w \cdot \begin{cases} CR(s_{\min}, s_{\max}, u) - r_{\max} & \text{for} \quad CR(s_{\min}, s_{\max}, u) > r_{\max} \\ r_{\min} - CR(s_{\min}, s_{\max}, u) & \text{for} \quad CR(s_{\min}, s_{\max}, u) < r_{\min} \\ 0 & \text{for} \quad r_{\min} \leq CR(s_{\min}, s_{\max}, u) \leq r_{\max} \end{cases} \tag{2.4}$$

2.4. Solver

The main difficulty of this problem is the fact that the objective value is indirectly dependent on the input sequence. First the input sequence is used to calculate all model values, such as the temperature over the strip length or the transformation fraction, which are used consecutively to compute the objective value of the control sequence. Hence the problem can be characterized as a black box, where in practice nothing useful is known about the complete construction of the objective value. Besides it is known that the physical model is non-linear and non-convex, which makes it extremely hard to find an exact solution for the problem.

The solver used to compute a setup, currently STORM, tries to find a control sequence $u(i)$ for $i \in \{1,\dots,n\}$, with n equal to the amount of cooling banks and $u(i) \in \{0,1,2\}$. Here $u(i) = 0$ means that bank i is switched off, $u(i) = 1$ means it is turned to half of its capacity or low flow and $u(i) = 2$ stands for full capacity or high flow. The search space of possible control sequences can be confined by constraints, these constraints are posed by physical properties of the setup, the defined boundary and the domain values of a problem instance as discussed in the previous sections.

Optimization for every bank individually is costly because of the large search space with a size of 3^{124} and may lead to very irregular patterns, which is undesirable because experience shows that the quality of products is higher and more constant when working with regular patterns. Therefore, this soft constraint is incorporated in the problem definition by choosing a more compact way to describe a solution. This representation uses only three variables to describe the pattern in one cooling zone: the length of the pattern, the load over the top sections and the load over the bottom sections over that length. With a cooling zone, a selection of consecutive banks that is responsible for one cooling rate is meant, such as described in Figure 2.2. Depending on the maximum allowed pattern length l, this results in a search space of $\mathcal{O}(l!^d)$ for d domains without symmetry or flow level constraints. From the value of these three variables the setup for one cooling zone is constructed by a repetition of a pattern with length equal to the value of the pattern length, and the sum of the intensities on top and bottom equal to the respective load values. Take for example the set of variables $[6,3,8]$, this results in a pattern as such:

```
Pattern length = 6
Top load = 3
Bottom load = 8
Top pattern ← 101010 | 101010 | 101010 | ...
Bottom pattern ← 211211 | 211211 | 211211 | ...
```

3 Theory

Optimization problems are characterised by different properties and these properties in turn define the type of methods that are best suitable to solve them. The problem as described in chapter 2 is a non-linear, non-convex problem, which should be solved in very limited time as close to the optimal solution as possible. Therefore, the use of global approximation methods is preferred [11]. The current system at Tata Steel is based on a genetic algorithm, a solution that is known to perform well on this type of problems. More details on previous research can be found in subsection 3.1.3.

In order to understand the methodology in chapter 4, this chapter will explain the relevant theory. First section 3.1 will explain the general genetic algorithm, after which section 3.2 will elaborate on multi-objective genetic algorithms.

3.1. Genetic Algorithms

As was briefly mentioned in chapter 1, the current solution implemented at Tata Steel is a genetic algorithm. This section will first explain the basics of a genetic algorithm and then expand to the different variations of operators that were used in this research.

3.1.1. The Basics

Genetic Algorithms, abbreviated as GAs, belong to the larger group of evolutionary algorithms, which are as the name implies inspired by evolutionary processes in nature and were first introduced by Holland [7]. Specifically, genetic algorithms encode solutions in so called *individuals* with a corresponding *gene sequence*, where every gene represents a variable of the problem. As depicted in Figure 3.1, there thus exists a function that maps the gene sequence of an individual to the decision vector and the objective function subsequently maps the decision vector to the objective vector. In this particular case the gene sequence is the representation as defined in section 2.4 and the decision vector is the control sequence, that contains the intensities allowed during cooling.

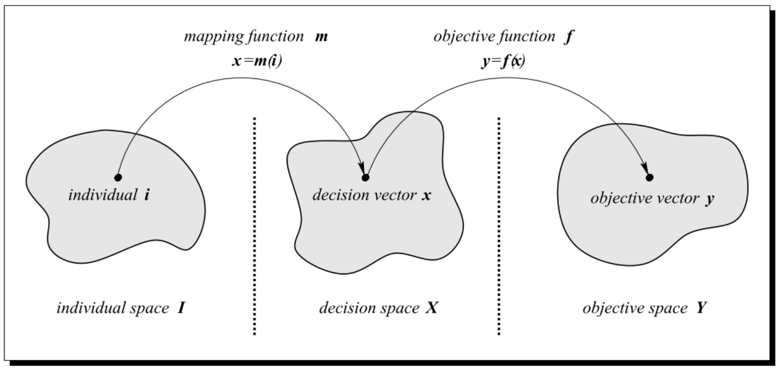


Figure 3.1: Relation between individual space, solution space and objective space

To initialize a genetic algorithm, a population is created of a predefined number of individuals. Generic pseudo-code of a genetic algorithms is listed in algorithm 1. The functions that are called in the algorithm define the different stages of the GA and can be varied according to the problem to solve. First each stage will be described and in the next section the possible variations of the so called operators relevant in this thesis are explained.

Algorithm 1 Genetic Algorithm

```
$t \leftarrow 0$
$P(t=0) \leftarrow$ Initial population
EVALUATE($P(t=0)$)
while not termination do
    $P_c(t) \leftarrow$ REPRODUCE($P(t)$)
    $P_c(t) \leftarrow$ MUTATE($P_c(t)$)
    $P_c(t) \leftarrow$ REPAIR($P_c(t)$)
    EVALUATE($P_c(t)$)
    $P(t+1) \leftarrow$ SELECT($P(t)$, $P_c(t)$)
    $t \leftarrow t+1$
return BESTINDIVIDUAL($P(t)$)
```

Initialization
In this stage the first individuals are formed, each of them represented to the chosen gene representation and applying to possible constraints in the problem definition. The first generation is often referred to as generation zero or the starting population.

Reproduction
Reproduction stands for combining two individuals, the *parents*, to obtain new individuals or *children*. This means that a subset of genes is selected from one parent and the complementary subset of genes from the other parent, to combine into one child and all the remaining genes into the other child. Reproduction enables the algorithm to explore different parts of the search space.

Mutation
In order to be able to explore the space close to a given solution, mutation is applied with a certain probability. Meaning that one, or possibly more, of the genes of an individual are changed to a different value within the domain. Besides, this is the only possibility to introduce gene values into the population that were not or are no longer present in it.

Reparation
Often it is not possible to choose genetic operators that respect all the constraints posed by the problem definition. Therefore, it might be necessary to repair the solutions generated before they can be evaluated: every new solution has to be checked for validity and if it is not valid a repair operation should turn it into a valid solution before evaluation.

Evaluation
In the evaluation phase all solutions are mapped to the corresponding decision vector and evaluated according to the defined objective function. These values are then stored, such that they can be used in to select the next generation.

Selection
The final step of each iteration consists of choosing the next generation based on all parents and their offspring. In choosing the most suitable selection operator, there is a trade-off between convergence and diversity. On one hand, fast convergence is desirable since a solution will be found fast. But on the other hand, it is important to maintain diversity in order to explore large parts of the search space to decrease the probability of termination in a local optimum.

3.1.2. Variations

Now that the outline of a genetic algorithm is clear, some ways to implement the different stages can be elaborated. As mentioned earlier, the choice of an operator depends on the problem at hand and is not always clear beforehand. Therefore testing the impact of the operators is often useful to make the right choices.

Initialization

When there is no prior knowledge of the search space or the location of the optimal solution, the only way to initialize the population is by randomly constructing individuals that comply with all constraints. When there is prior knowledge, it is desirable to construct a population that can use this knowledge to speed up the search, thereby having a so called *warm start*. In the current implementation, this is done by adding the optimal solution of a previous computation with similar parameters to the initial population. Similar meaning a steel slab cooled with the same recipe and a thickness, finishing temperature and maximum speed close to the values of the current slab.

The extensions developed in this research aim to find a good starting point for the computation of sample setups. Since the samples are all related to the base setup, currently used by the optimizer, this is used to determine which are promising individuals. Exactly how these individuals are found will be discussed in the next chapter.

Reproduce

In all implementations in this research the simplest form of reproduction is used: *single-point crossover*. There is no need to complicate this, since only a few variables describe the problem thus the probability of finding the right crossover point is very high. For clarity the following uses a larger gene sequence. Suppose two parent individuals can be represented as follows:

$$\begin{aligned} P_1 &= \quad 1111200200000000221111 \\ P_2 &= \quad 2121212121000022112211. \end{aligned} \tag{3.1}$$

A random number c is generated and both gene sequences are split at that particular point:

$$\begin{aligned} P_1 &= \quad 11112002000 \quad 00000221111 \\ P_2 &= \quad 21212121210 \quad 00022112211. \end{aligned} \tag{3.2}$$

Now two new individuals, the children, are created by swapping the last part of the first parent with that of the second parent:

$$\begin{aligned} C_1 &= \quad 11112002000 \quad 00022112211 \\ C_2 &= \quad 21212121210 \quad 00000221111. \end{aligned} \tag{3.3}$$

Mutation

On each new individual mutation is applied. This means that again a random number m is generated and the gene at that index is replaced with a randomly generated number within the domain of that gene. The probability that the gene is permuted is therefore $\frac{1}{max\_value}$, where max_value is the maximum feasible value of that particular gene, since it is possible that the value generated is equal to the value that was there in the first place and therefore no change occurs.

Reparation

The repair operator for this particular problem has to check two things:

1. Are both densities of a zone feasible given the length of the pattern to be constructed? If not a new value is randomly selected within the feasible range.

2. Are the symmetry constraints still respected? In case of flow symmetry one of the densities is selected and changed to the value of the other density and in case of section symmetry a value within the feasible range is randomly chosen.

Selection

The easiest selection method is to sort the population on fitness and simply choose the the ones with the highest fitness value, this method is called *fitness* selection. Because this can lead to loss of diversity, other methods can also be used for selection. Some of these methods are described in [18]. Only tournament selection, one of the most commonly used selection methods, was added to the used selection methods to demonstrate the influence of having another selection method. The choice for tournament selection was made based on the experiment described in Appendix A.

In tournament selection, all parents and children are divided into pairs and the best of each pair goes to the next round, until enough individuals have been selected and the new population is formed. This technique helps to maintain diversity in the population, at the cost of some convergence speed compared to fitness selection.

3.1.3. Applications of Genetic Algorithms

Genetic algorithms have been applied successfully in a wide variety of fields from control strategies for electric vehicles [14] to adaptive design optimization in wireless networks [4] and from combined location-inventory-routing problems [6] to supply chain optimization [1]. All of these applications have in common that they try to find a globally optimal or near optimal solution to a problem that is often NP-hard. GAs are praised for their fast operation and their ability to avoid stranding in a local optimum even in large non-linear and non-convex search spaces, such as the physical model used by Tata Steel to find an optimal cooling pattern, where exact solvers are no longer of use.

Further, the concept of using an informed initialization or warm start has been used to enhance the performance of different algorithmic solutions such as deep neural networks [8], mixed-integer programs [13] and Bayesian optimization programs [9]. In each of these applications the algorithms learn from either data sets on which they are trained or from solving problem instances. Without the addition of a warm start, this information can get lost in successive runs and therefore cause longer run times. When this information is passed on through informed initialization, it can be exploited to speed up the learning process and convergence of the solver significantly.

This research combines the use of a genetic algorithm and informed initialization to explore the potential of the GA as a solution for future application in more advanced cooling path optimization. Dasgupta et al. [2] already showed that informed initialization has a promising effect when used in combination with multi-objective genetic algorithms, which is cause to believe that the same holds for the single-objective implementation.

3.2. Multi-objective Genetic Algorithms

Currently, the problem is approached with a single-objective solution. In order to obtain one single objective, the value of multiple objectives is summed while weighing each value with a user defined weight to possibly emphasise some objectives more than others. Another way to approach this would be using a multi-objective algorithm. Algorithms of this category aim to find a group of non dominating solutions, which can be used as a starting population for sample setup computations. One of the most well known multi-objective genetic algorithms is the Non dominated Search Genetic Algorithm II (NSGA-II) as introduced in Deb et al. [3]. The algorithm does not require any parameter specification and is easy to implement and is therefore the perfect candidate solution for the purpose of investigating the influence of using a multi-objective genetic algorithm to construct generation zero.

3.2.1. Foundation

Before we can understand how NSGA-II works, there are two things that need explanation first: the non dominated sorting approach and the method used for diversity preservation. Thereafter, subsection 3.2.2 introduces the algorithm.

Non Dominated Sorting

This procedure is based on the concept of domination: a solution is said to dominate another solution when it performs better in at least one of the objectives and equally good in all other objectives. During the sorting process the solution will be divided into *fronts*. The first front $\mathcal{F}_1$ will contain the solutions that are not dominated by any other solution in the population, the second front $\mathcal{F}_2$ the solutions that are not dominated by any other solutions after removing the ones in the first front, etc. Here the subscript of the front a solution is in, is equal to the *rank* of the solutions it holds. The pseudocode of the non dominating sort function as performed in NSGA-II can be found in Algorithm 2.

Algorithm 2 Fast Nondominated Sort

```
function FASTNONDOMINATEDSORT($P$)
    for all $p \in P$ do
        $S_p \leftarrow \emptyset$
        $n_p \leftarrow 0$
        for all $q \in P$ do
            if $p \prec q$ then                                    ▷ If $p$ dominates $q$
                $S_p \leftarrow S_p \cup q$
            else if $q \prec p$ then
                $n_p \leftarrow n_p + 1$
        if $n_p = 0$ then                                          ▷ $p$ belongs to the first front
            $rank_p \leftarrow 1$
            $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup p$
    $i \leftarrow 1$
    while $\mathcal{F}_i \neq \emptyset$ do
        $Q \leftarrow \emptyset$                                   ▷ Used to store members of the next front
        for all $p \in \mathcal{F}_i$ do
            for all $q \in S_p$ do
                $n_q \leftarrow n_q - 1$
                if $n_q = 0$ then                                  ▷ $q$ belongs to the next front
                    $rank_q \leftarrow i + 1$
                    $Q \leftarrow Q \cup q$
        $i \leftarrow i + 1$
        $\mathcal{F}_i \leftarrow Q$
    return $\mathcal{F}$
```

Diversity Preservation

Besides having the best solutions, it is also important to preserve diversity within the population. NSGA-II in particular uses a crowded-comparison approach to achieve this. To estimate the solution density around a particular solution in the population, the average distance of two points on either side of this point along each of the objectives is calculated. This quantity is called the *crowding distance* in a particular objective. In each objective the solutions with the highest and lowest value, which therefore only have one neighbour in this projection, are assigned infinite crowding distance. Doing this makes sure that these extremes will always survive to the next population. To compute the overall crowding distance, the crowding distances for every objective are summed. Algorithm 3 outlines the computation procedure of the crowding-distances for all solutions in the set $\mathcal{I}$.

The distance metric allows the algorithm to compare two solutions with respect to their proximity to each other. A solution with a smaller crowding distance is more ’crowded’ by other solutions and therefore gives less new information about the front and thus the crowded comparison operator $\prec_n$, that guides the selection process of the algorithm towards a spread-out Pareto-optimal front, can be defined as follows:

$$i \prec_n j \quad \begin{array}{l} \text{if } i_{\text{rank}} < j_{\text{rank}} \\ \text{or } i_{\text{rank}} = j_{\text{rank}} \text{ and } i_{\text{distance}} > j_{\text{distance}} \end{array} \tag{3.4}$$

In words, when comparing two solutions, the solution with lower rank is preferred over the other. If the two solutions have equal rank, the one with the highest crowding distance and thus located in a lesser crowded region of the front, is superior.

3.2.2. The Algorithm

To construct a population of diverse individuals close to the Pareto front, NSGA-II goes trough an initialization phase and an iteration phase similar to the general genetic algorithm. In the initialization phase a random population of size N and offspring are created. Offspring is constructed by selection, crossover and

Algorithm 3 Crowding Distance Assignment

```
procedure CrowdingDistanceAssignment($\mathcal{I}$)
    $l \leftarrow |\mathcal{I}|$
    for $i \leq l$ do
        $\mathcal{I}[i]_{distance} \leftarrow 0$
    for all objectives $m$ do
        $\mathcal{I} \leftarrow \text{sort}(\mathcal{I}, m)$                          ▷ Sort by each objective value
        $\mathcal{I}[1] \leftarrow \infty$ and $\mathcal{I}[l] \leftarrow \infty$      ▷ Such that boundary points are always selected
        for $i = 2$ to $l-1$ do                                                         ▷ All other points
            $\mathcal{I}[i]_{distance} \leftarrow \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{max} - f_m^{min})$
```

mutation, where the selection method used is tournament selection followed by single-point crossover or random mutation. Then the algorithm enters the iteration phase until the termination criteria are met. The pseudocode is listed in Algorithm 4.

Algorithm 4 NSGA-II

```
$P_0 \leftarrow$ Random parent population
$Q_0 \leftarrow$ MakeNewPopulation($P_0$)
while not termination do
    $R_t \leftarrow P_t \cup Q_t$
    $\mathcal{F} \leftarrow$ FastNonDominatedSort($R_t$)
    $P_t \leftarrow \emptyset$, $i \leftarrow 1$
    while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ do
        CrowdingDistanceAssignment($\mathcal{F}_i$)
        $P_{t+1} = \leftarrow P_{t+1} \cup \mathcal{F}_i$
        $i \leftarrow i + 1$
    Sort($\mathcal{F}_i, \prec_n$)
    $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|]$
    $Q_{t+1} \leftarrow$ MakeNewPopulation($P_{t+1}$)
    $t \leftarrow t + 1$
return $P_t$
```

3.2.3. Applications of NSGA-II

The idea of using a multi-objective genetic algorithm to explore the search space to find candidate solutions has been used in various fields such as the design of airfoils for a Mars airplane propeller blade [15] and identifying genes associated with the identification of cancer cells [17]. The difference with the currently examined problem is that in both cases it is impossible to choose which of the candidate solutions is the best without an expert inspecting them, where in choosing the best pattern is straightforward since it is the one leading to the cooling path closest to optimal. Therefore, in the case of cooling setup optimization considered in this thesis, the candidate solutions can be used as input for a second algorithm that is able to use them as a starting point to find the optimal solution of a particular problem instance.

This can be explained by the fact that the objectives of a single setup are not contradictory and can therefore be summarized in a single objective function, whereas the problems that leave experts with the deciding vote have two or more objectives that can never both have their optimal value in a valid solution. An example of such a problem can be found in [10], where there are two counterproductive objectives. Despite the optimization problem being in essence a single-objective, using a multi-objective algorithm to obtain candidate solutions is assumed to lead to faster convergence because this allows for creating an elite pool based on a bigger set of objectives which can be stored and the proper individuals can be selected for warm start as soon as the true objective set is known.

4 Methodology

The goal of this research is to examine possible improvements to the current system that is used for the cooling process after hot rolling at Tata Steel and determine whether the solver can be used to accommodate for more complex products. To that extend, three different extensions will be introduced: STORM-TSM is discussed in section 4.2 and the multi-objective based solutions STORM-TMM and STORM-TMO are discussed in section 4.3. Before diving into the newly introduced methods, section 4.1 will state the outline of the current solver. In section 4.4 the problem instances used in the performed experiments are described and, lastly, section 4.5 describes the methods used to test the statistical significance of the results.

4.1. Current Implementation

The cooling system at Tata Steel is driven by a simple genetic algorithm, incorporated in the full control system STORM. Computation of a new setup is initialized with a random population, with the exception that the best individual found for a previously encountered steel slab with similar variables is added to it. Then single-point crossover and random mutation are applied, to construct the next generation by picking the fittest individuals of the combined set of parents and offspring. Note that for every slab only one setup is computed, tailored to the estimated maximum speed during the cooling process and the finishing temperature measured at the head of the slab.

The found setup defines which banks the controller is allowed to use during the cooling process. To determine which banks should be switched at what time, the controller uses so called *xtv*-tables that contain the static relation between place, time and speed of a sample and a feedback mechanism to compensate for temperature differences. In order to further automate the cooling process, faster computation of setups is required. This way it should be possible to compute a custom setup for each steel sample, which can be sent directly to the controller, directly limiting the need for additional feedback mechanisms. This research proposes to use an informed start to speed up the process, as mentioned in subsection 3.1.3 and subsection 3.2.3. Three different methods to implement this will be introduced in the upcoming sections, all require a precomputation which can be finished before the steel slab enters the cooling system with estimated parameters.

It would have been preferred to use an implementation of STORM as the baseline to compare the newly developed algorithms to, but unfortunately it was not possible to work with the original system and the output values of both implementations do not align. This results in that the promising individual found in historical data is not a promising solution in the implementation created for the conducted experiments. Therefore, the choice has been made to use an uninformed implementation (STORM-U) as the baseline.

4.2. Single Model Method

The starting population of the current solution as discussed in the previous section, is already a little informed by adding one likely well fitted individual to it from a previous computation of for a similar slab. This research extends this idea by running an additional computation to create generation zero, which can be done before the slab enters the finishing mill with approximated parameters. This results in the method called STORM-Trained on a Single Model (STORM-TSM), since the population is trained with the physical model with one set of parameters and with one set of objectives. The quality of this population, which can be referred to as an *elite set*, is defined by two things:

1. the proximity of the best solution in the population to the optimal value of the problem instance and
2. the diversity within the population.

The first is important since we know the variables of the samples will be similar to the problem instance used to construct the elite set and their solutions will therefore most likely be similar. The second measure is there because starting with a population consisting of many similar or even equal individuals, increases the chance of convergence to a local optimum.

There are two main properties of the genetic algorithm that influence the convergence and the diversity of the population, which are the population size and the selection operator. Since computation time is limited, the choice has been made to keep the population size at 20, as it is in STORM. Further, the two most commonly used selection operators (fitness selection and tournament selection) are considered. The influence of taking a particularly constructed elite set on the performance of the solver for different samples was assessed by examining the number of iterations until convergence and the proximity of the attained value to the optimal value.

Since it is not possible to determine the exact optimal value of the problem instances, a reference value is defined for each of the instances. This is done by trying random solutions for an hour per instance, which comes down to about 10^6 iterations, and storing the best solution. Due to the fact that enough solutions are examined this should be close to the true optimal value.

4.3. Multi Model Methods

Another method to create a good starting point for sample setup computation is based on a multi-objective genetic algorithm (MOGA). In this research the choice has been made to work with NSGA-II in particular, as introduced in subsection 3.2.2, since this is a well-known algorithm with excellent performance on proximity to the Pareto front and diversity and is easy to implement as well. In this research two different multi-objective extensions are examined. The first being STORM-Trained on Multiple Models (STORM-TMM), where the different objectives in the precomputation are the objective values of the physical simulation with different values for the estimated maximum speed and the finishing temperature of the slab. The second extension is STORM-Trained on Multiple Objectives (STORM-TMO), which trains the population based on a large set of objectives with trade-offs among them, where each sample for which a setup is to be constructed has to comply to a subset of these objectives.

In both methods the population size in precomputation can be bigger than that of the final computation, in this case the objective values of the informed population are computed for the given sample and the fittest individuals are chosen to form generation zero. The performance of each of the MOGA solutions is compared to the performance of STORM-U and that of STORM-TSM.

4.4. Problem Instances

To construct the problem instances consisting of multiple samples of a steel slab, the historical data of four steel slabs was used. Each of the steel slabs was cooled according to its own recipe, two of which have a single domain and two of which have a double domain. Hence the cooling pattern of the slabs can be described with respectively six and twelve parameters. The domains and objectives of the recipes are summarized in Table 4.1 and Table 4.2 respectively. Note that some of the minimum and maximum values of the objectives are redacted, due to their sensitivity.

#	Type	Start	End	Symmetry	Flow top	Flow bottom
1	Main	1	54	Section	Mixed	Mixed
2	Main	1	54	Section	High	High
	Main	55	62	Section	High	High
3	Main	1	54	Section	Mixed	Mixed
4	Main	1	54	Flow	Mixed	Mixed
	Main	55	62	Section	High	High

Table 4.1: Domain for all recipes

#	Type	Start	End	Minimum	Maximum	Weight	Recipe value	Transfer point
1	TempPnt	1	62	-5.0	5.0	1.0	IT	None
	CR	1	62	-	-	1.0	None	EndP
	CR	1	62	-	-	1.0	None	StartPP
	Temp	63	63	-5.0	5.0	5.0	CT	None
2	TempPnt	1	54	-	-	1.0	None	None
	CR	1	54	-	-	1.0	None	EndP
	CR	1	54	0.0	0.0	1.0	None	StartPP
	CR	55	62	0.0	0.0	1.0	None	None
	Temp	63	63	-5.0	5.0	5.0	CT	None
3	TempPnt	1	54	-	-	1.0	None	None
	CR	1	54	-	-	1.0	None	EndP
	CR	1	54	-	-	1.0	None	StartPP
	Temp	63	63	-5.0	5.0	5.0	CT	None
4	TempPnt	1	54	-	-	1.0	None	None
	CR	1	54	0.0	0.0	1.0	None	StartPP
	CR	55	62	0.0	0.0	1.0	None	None
	Temp	63	63	-5.0	5.0	5.0	CT	None

Table 4.2: Objectives for all recipes

4.4.1. Sample creation

In the experiments conducted, two different experiments can be distinguished with different methods to construct samples based on the historical data described in the previous section. The first set of experiments tests the performance of STORM-TSM and STORM-TMM. Here the estimated maximum speed of the slab and the finishing temperature of the head are varied, thereby creating samples that aspire the same goals at the end of the run-out table while having a slightly altered optimal cooling path. The values used for the simulations used to train the population with STORM-TMM and for the test samples are listed in Table 4.3, where each combination of velocity and temperature is used for the given purpose. The second experiment again includes STORM-TSM and compares it to STORM-TMO. Now all physical variables remain constant, but the target values of the objectives are varied and phase fraction objectives are added. Each sample aims to satisfy a different set of these objectives, where each set contains one temperature point objective, one temperature objective, two cooling rate objectives and transformation fraction objectives adding up to 100%. The additional objectives are listed in Table 4.4. These objectives are strictly made up for the purpose of this research. Then setups were computed for samples that each had a subset of these objectives.

#	Application	Variable	Values
1	Informed population	Maximum velocity	12.5, 13, 13.5, 14.5, 15
		Finishing temperature	979, 985, 987, 989, 992
	Sample setup	Maximum velocity	12.7, 13.1, 14.6
		Finishing temperature	980, 984, 990
2	Informed population	Maximum velocity	12.9, 13.4, 13.9, 14.9, 15.4
		Finishing temperature	900, 815, 830, 845, 960
	Sample setup	Maximum velocity	13.2, 13.7, 15.1
		Finishing temperature	820, 850, 920
3	Informed population	Maximum velocity	2.5, 2.9, 3.3, 3.7, 4.1
		Finishing temperature	870, 880, 890, 900, 910
	Sample setup	Maximum velocity	2.7, 3.1, 3.5
		Finishing temperature	878, 892, 905
4	Informed population	Maximum velocity	2.5, 2.9, 3.3, 3.7, 4.1
		Finishing temperature	870, 880, 890, 900, 910
	Sample setup	Maximum velocity	2.7, 3.1, 3.5
		Finishing temperature	878, 892, 905

Table 4.3: Values of model variables for training and samples for all recipes

#	Type	Start	End	Minimum	Maximum	Weight	Recipe value	Transfer point
1	TempPnt	1	62	645	655	1.0	None	None
	Temp	63	63	705	715	5.0	None	None
	Ferrite	32	32	98	102	1.0	None	None
	Ferrite	63	63	88	92	1.0	None	None
	Perlite	63	63	8	12	1.0	None	None
	Ferrite	63	63	98	102	1.0	None	None
2	TempPnt	1	54	758.0	762.0	1.0	None	None
	CR	1	54	20.0	30.0	1.0	None	None
	CR	55	52	20.0	30.0	1.0	None	None
	Ferrite	32	32	98	102	1.0	None	None
	Ferrite	63	63	88	92	1.0	None	None
	Perlite	63	63	8	12	1.0	None	None
	Ferrite	63	63	98	102	1.0	None	None
3	TempPnt	1	54	964	974	1.0	None	None
	Temp	63	63	655	665	5.0	None	None
	Ferrite	32	32	98	102	1.0	None	None
	Ferrite	63	63	88	92	1.0	None	None
	Perlite	63	63	8	12	1.0	None	None
	Ferrite	63	63	98	102	1.0	None	None
4	TempPnt	1	54	845	855	1.0	None	None
	Temp	63	63	460	470	5.0	None	None
	Ferrite	32	32	98	102	1.0	None	None
	Ferrite	63	63	88	92	1.0	None	None
	Perlite	63	63	8	12	1.0	None	None
	Ferrite	63	63	98	102	1.0	None	None

Table 4.4: Additional objectives for all recipes

4.5. Statistical Significance

Finally, this section will dive into the importance of statistical significance and how to ascribe a value to it. In everyday language something that is significant is often large or important, though in statistics this is not necessarily so. A result in the acquired data is statistically significant when it is very unlikely that it is there by chance and can therefore be used to confirm or reject a stated hypothesis. This is a most relevant part of this research, because the outcomes are based on experimental data.

After gathering the outcomes of different algorithmic implementations, such as the number of iterations until convergence and the objective values obtained, these values can be compared to each other. In order to be able to base conclusions on these comparisons, it is required that the significance of these results be tested. The most common test to use for this purpose is Student's t-test [12], where the t stands for the t-statistic whose numerical value is proportional to the probability of the difference between means to be statistically significant. It is defined as

$$t = \frac{\mu_1 - \mu_2}{\sigma\sqrt{\frac{1}{n_1} - \frac{1}{n_2}}}, \tag{4.1}$$

where μ_i are the mean values of the compared groups, n_i the number of samples in each of them and σ their shared standard deviation. A larger t-value indicates that the difference between means is more likely to be significant, which is achieved by either showing a large difference in means or having many samples.

One of the main assumptions of the t-test is that the variation in the compared groups is equal, which is not true in the results gathered to support this research. Many adaptations to the t-test have been defined to adjust for data where these assumptions do not hold, see [20] for an overview. To assess the significance of the results discussed in this thesis, the independent samples t-test will be used, designed to compare multiple groups of samples that have no correlation between them [5]. Here the t-statistic is defined as

$$t = \frac{\mu_1 - \mu_2}{MSE/\sqrt{n}}, \tag{4.2}$$

where MSE is the Mean Square Error of the groups to be compared and is defined as the sum of the variances divided by N, the number of groups:

$$MSE = \frac{\sum_{i=1}^{N} \sigma_i}{N}. \tag{4.3}$$

Notice that here the assumption is made that each group consists of the same amount of data points, equal to n. With the t-statistic, the difference between two means and the number of degrees of freedom, the probability p that the difference between the two means is there by chance can be determined with a statistic tool, available in most programming languages and analysis programs. In this case the Python statistical method `stats.t.cdf(x,df)`, from the SciPy package is used [19]. The number of degrees of freedom is equal to $N(n-1)$, the total number of observations minus the number of groups that are compared. There is no exact definition of the threshold for when results are statistical significant, though it is customary to say that a hypothesis can be assessed when the probability of difference caused by chance is lower than 5%.

5 Experiments

According to the methodology described in chapter 4, various experiments were conducted to investigate which of the algorithms is able to compute setups for different steel samples within limited time. First, section 5.1 outlines the general setup of an experiment as performed to compare the performance of different methods. The experiments can be divided into two categories, section 5.2 describes the comparison of STORM-TSM and STORM-TMM to each other and STORM-U, where the samples differ based on the values of the estimated maximum velocity and the finishing temperature. Then, section 5.3 analyses the influence of changing the objectives for different samples on the performance of STORM-TSM and STORM-TMO compared to STORM-U.

5.1. Experimental Setup

To show that tuning the starting population by precomputation has a positive influence on the computation time, sample setups were computed starting from a random population and starting from a tuned population. The tuned population being constructed with one of the methods as introduced in chapter 4. To account for statistical influences each experiment was run 50 times for each set of parameters, the parameters being the selection method (either fitness selection and tournament selection) and the precomputation method (STORM-TSM, STORM-TMM or STORM-TMO and STORM-U).

The number of iterations was limited at 150 for recipe 1 and 3 and at 400 for recipe 2 and 4. Since the latter have multiple domains and thus require more variables to describe the setup, in some cases more iterations were needed to demonstrate the difference between different settings. It is expected that the algorithm converges faster when it has a warm start and possibly finds values closer to optimal, if it better able to avoid local minima. The implementations are compared based on two quality measures:

- The fitness of the optimal setup found by the algorithm
- And the amount of iterations required to obtain a *useful* solution, where a solution is assumed to be good enough when it lies within 20% of the reference value.

The fitness of a setup is expressed as a measure of proximity of the best found value to optimal defined as

$$V = \frac{f_{GA} - f_r}{f_r}, \tag{5.1}$$

where f_r is the reference value as introduced in section 4.2. This measure allows comparison between the results for different samples, since these do not have the same optimal value. Also this representation directly shows how many of the solution are useful, since a setup is useful when it has a relative value of 0.2 or lower.

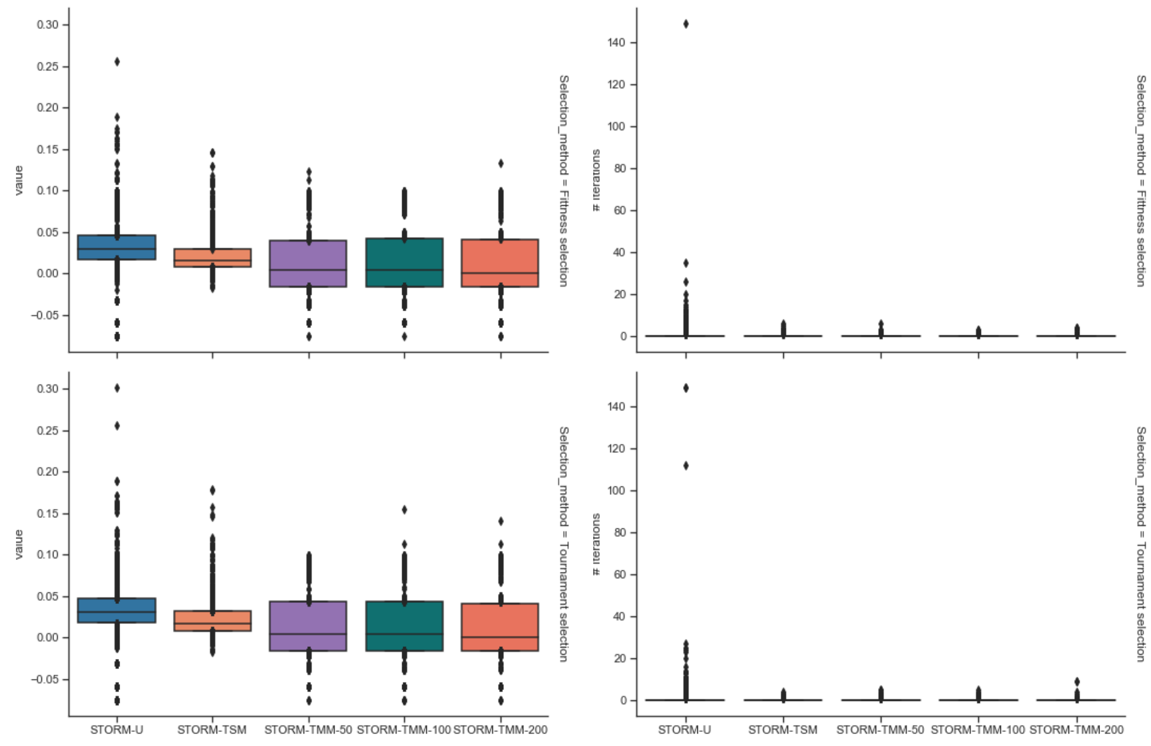
All graphs showing results are so called box-plots. A box plot consists of a box summarizing 50% of the data. The data points that lie outside of the range containing half of the data points are plotted separately. Often these are summarized by lines reaching multiple standard deviations, but since the distribution of the points is unknown and most likely not normal this could be misleading. Finally, the line within the box represents the median.

5.2. Changing Variables

In the first experiment a starting population is created based on a single model with STORM-TSM and based on 25 different models with STORM-TMM, one for every combination of maximum velocities and finishing temperatures as listed in section 4.4. Samples are then created to have parameters somewhere in the same interval, also listed in said table. The population size for precomputation with STORM-TMM is taken to be 50, 100 and 200 and for STORM-TSM to be 20 as for STORM-U. Finally, the selection operator is chosen to be either fitness selection or tournament selection for computation of the sample setups with a population of size 20.

The results for recipes with one domain are depicted in Figure 5.1 and show that STORM-TMM guarantees the best optimal values. Statistical analysis shows that from the data it is not possible to decide which of the selection method or value of the population size performs best. The outcomes of the statistical analysis are listed in Table B.1 in Appendix B, where the relevant rows are highlighted. This is most likely due to the fact that the algorithm converges fast to close to optimal and then still has enough time for further improvement. Also the number of iterations required to obtain a usable solution is always smaller than 20 iterations for STORM-TMM, though it does not outperform STORM-TSM. Now, Figure 5.2 summarizes the results for the recipes with two domains and here STORM-TSM with a population size of 200 clearly outperforms all the other algorithms. It is not clear why this population size has such a positive influence on the solution quality and still the solution found is often not good enough to be used, so more improvement is needed to speed up the search.

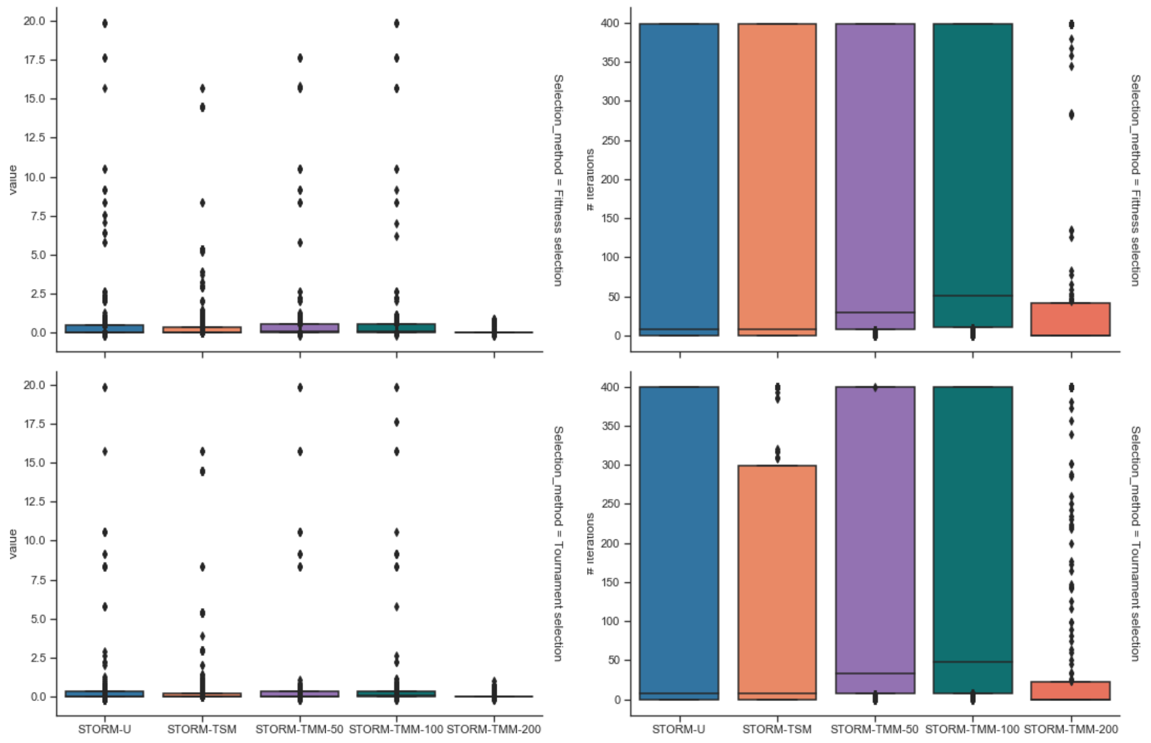
Based on the results as depicted here and the statistical analysis performed, of which the results are to be found in section B.1, it can be said that training on multiple models is the best way to warm start STORM for all recipes tested. When a recipe has only one domain, a population size of 50 or possibly lower is sufficient for precomputation, though for multiple domains a population of definitely more than 100 individuals is required.



(a) Solution quality

(b) Number of iterations required to obtain a usable solution

Figure 5.1: Effect of selection methods on number of iterations on both quality measures for recipes with one domain, over the length of the ROT for STORM-U, STORM-TSM and STORM-TMM.



(a) Solution quality

(b) Number of iterations required to obtain usable solution

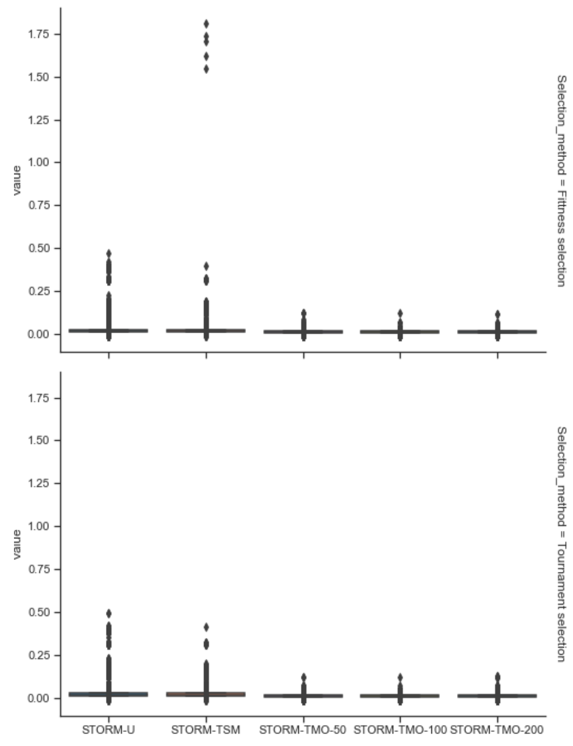
Figure 5.2: Effect of selection methods on number of iterations on both quality measures for recipes with two domains, over the length of the ROT for STORM-U, STORM-TSM and STORM-TMM with different MOGA population sizes.

5.3. Changing Objectives

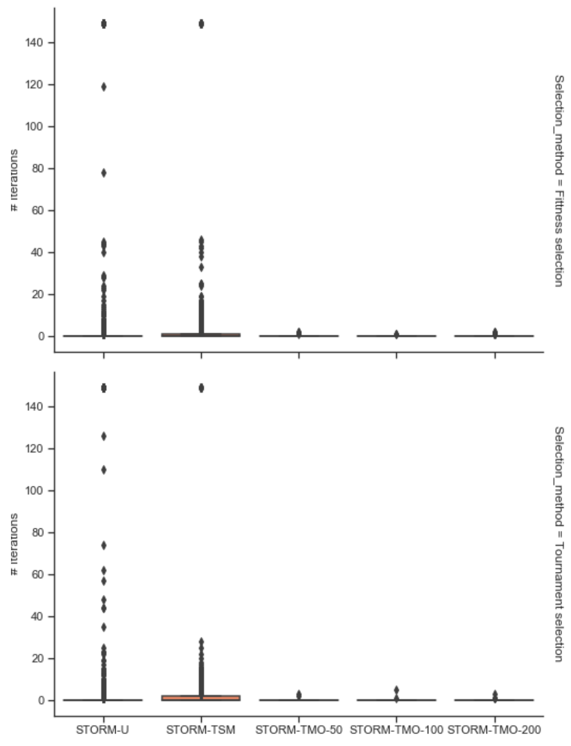
Another way to create different samples would be to not necessarily use variations in the incoming properties of the slab such as the estimated speed or the incoming temperature, but change the desired output properties. As discussed in chapter 2, these properties are a consequence of the goals of the cooling process which are translated in the objectives. The objectives used for these experiments are summarized in section 4.4. This representation of the samples underlines the fact that the objective function is made up of different objective functions, which can be exploited by using a multi-objective genetic algorithm.

For the experiments again a starting population is created based on a single model with STORM-TSM and with STORM-TMO a starting population is created based on the complete set of objectives for each recipe, as listed in section 4.4 and then samples are created that need to comply with a subset of these objectives. These subsets contain all feasible sets, where a set is considered feasible when there is one temperature point and one temperature objective, an equal amount of target cooling rates per domain as the original recipe and ferrite and perlite objectives that sum to 100% in each domain. Besides that, the experimental setup is the same as in the previous experiment.

The results for this experiment for recipes with one domain are depicted in Figure 5.3. Here it can be seen that especially when using STORM-TMO, a useful setup is often already present in the informed population created in precomputation and this method also ensures finding the lowest fitness values. The other two methods sometimes do not converge to a setup with a value lower than 1.2 times the reference value within the iteration limit. In combination with the results of the statistical analysis, as listed in Table B.5 and Table B.6 in Appendix B, it can be said that STORM-TMO outperforms both STORM-U and STORM-TSM for this purpose. The size of the population during precomputation with STORM-TMO does not have a noticeable effect on the performance of the algorithm. For recipes with two domains again the results are far less promising, as can be seen in Figure 5.4. All methods struggle to find useful setups within the iteration limit of 400, though STORM-TSM seems to have the least trouble and often holds a useful setup in generation zero. This could be due to the fact that there is more information in creating a setup for a feasible set of objectives, compared to a setup optimal for only one objective at a time as STORM-TMO does. Interesting to see is that the size of the population in precomputation with STORM-TMO does have an influence on the performance of the algorithm with respect to the number of iterations, where increasing the population leads to faster convergence on average. Also tournament selection appears to be the preferred selection method when optimizing for different sets of objectives.

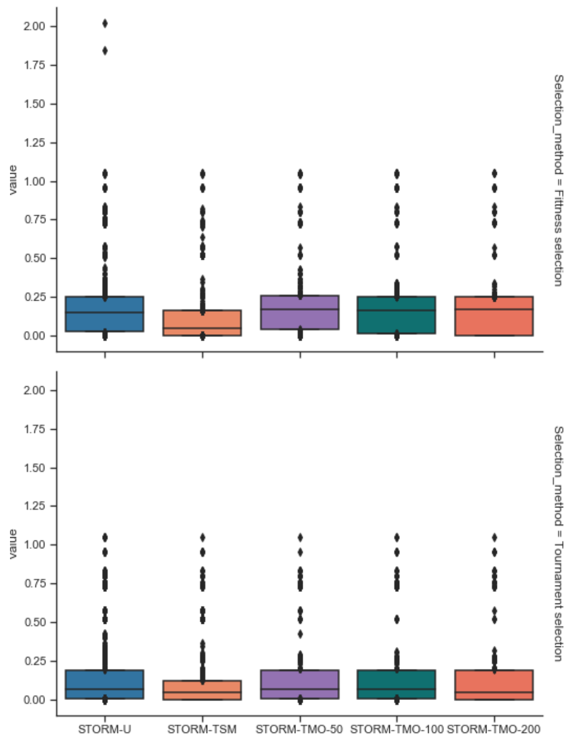


(a) Solution quality

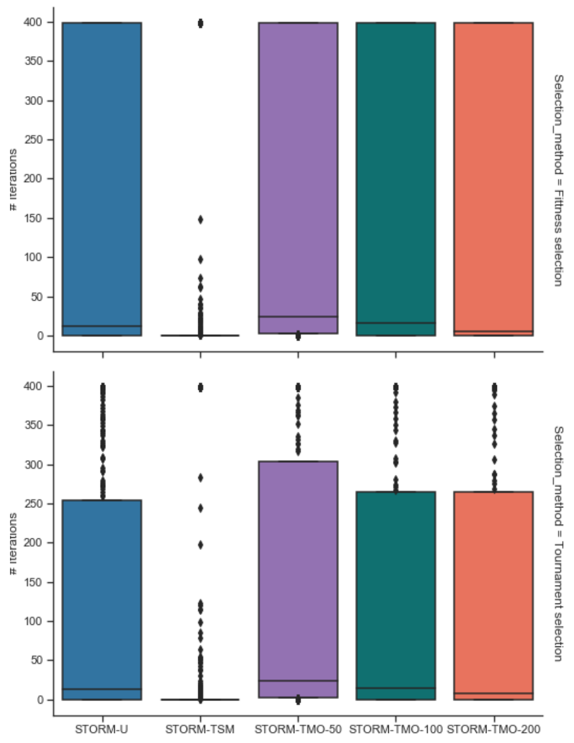


(b) Number of iterations required to obtain usable solution

Figure 5.3: Effect of selection methods on number of iterations on both quality measures for recipes with one domain, for different sets of objectives for STORM-U, STORM-TSM and STORM-TMM.



(a) Solution quality



(b) Number of iterations required to obtain usable solution

Figure 5.4: Effect of selection methods on number of iterations on both quality measures for recipes with one domain, for different sets of objectives for STORM-U, STORM-TSM and STORM-TMM.

5.4. Reflection

To summarize, the experiments have shown that using an informed starting point for setup computation with STORM has a positive effect on its performance. Creating generation zero is most effective with STORM-TSM or STORM-TMO. Both of these methods are based on a multi-objective genetic algorithm, where the objectives can be either physical models with different values for its simulation parameters or true objectives respectively. However, for recipes with two domains the results were not good enough. When optimizing over the length of the run-out table STORM-TSM performed best with a population size of 200. Optimization for different sets of objectives with two domains is best done by STORM-TSM. But all in all, for recipes with two domains neither of the methods investigated is sufficient to create a useful setup within 400 iterations.

There are some points of discussion that apply to all the experiments performed for this research. The main point on which one can criticize is the small number of problem instances used to validate the hypotheses. This is caused by the limited time in which the experiments had to be executed and the limited computer power available, mainly due to the limited MATLAB license which only allows running on one device at a time. In order to validate the methods investigated, more experiments will have to be completed.

Another remark is the fact that it is impossible to compare the outcomes discussed here with historical data, since computation of the objective value of historical patterns results in values that are far from optimal. This may originate from different parts of the implementation. First of all, the model used for these experiments is not the exact model as implemented in the cooling system in the finishing mill. Besides there could be discrepancies in the calculation of the objective function or the exact parameter values of the cooling installation.

Finally, it did not fit within the available time to examine the control actions required to switch between patterns of different samples. Remember that it is desirable to keep the amount of control actions to a minimum and therefore the usefulness of the discussed methods in the cooling system is not yet proven.

6 Conclusion

To recapitulate, the aim of this research was to explore and extend the current implementation used by Tata Steel IJmuiden to compute a cooling setup for the installation in the finishing mill, after hot rolling steel, in order to see whether it is fit for the challenges coming ahead. The search was guided by the following three research questions, as defined in chapter 1:

1. What characteristics influence the complexity of the problem?
2. Is the current method suitable to solve this particular problem and future applications?
3. Can the current method be extended to solve the problem more efficient?

This section will summarise the conducted experiments and answer these questions based on the conclusions that can be drawn from the results.

First, in chapter 2 a comprehensive description of the problem and its difficulties was given. The complexity of the problem is mainly caused by the indirect nature of the calculation of the objective value: based on the proper parameter values (such as temperature, speed and grain size) and a cooling pattern a physical simulation is run supplying a set of output values. These values are in turn used to compute the objective function value, assigning a fitness value to the cooling pattern. One approach could be to simplify and approximate this model, but since it is non-linear and non-convex this is unlikely to give satisfying results. Therefore, the problem as a whole is considered a black box. Combined with the fact that the solution does not have to be the exact optimum, but at least a good approximation to the global optimum a genetic algorithm as is already in place is a justified solution. However, extension of the algorithm is required to speed up the computation process before it can be used to compute multiple setups within the limited time available. Therefore, three methods have been introduced in chapter 4: STORM Trained on a Single Model, STORM Trained on Multiple Models and STORM Trained on Multiple Objectives. All of these methods perform a precomputation, as it were training the population to create a warm start and induce faster convergence.

The experiments as described in chapter 5, perform tests for two different purposes. The first being optimization over the length of the ROT, where each sample differs based on its estimated maximum speed and finishing temperature. Here STORM-U, STORM-TSM and STORM-TMM were tested and the results, which can be found in section 5.2, showed that STORM-TMM finds the best setups in all cases. However, for instances with multiple domains a population size of 200 for precomputation is required for good performance and even then not always useful setups are found. The second purpose for which STORM could be used in the future, is optimization for different objectives. See section 5.3 for the detailed comparison of the performance of STORM-U, STORM-TSM and STORM-TMO. In this environment there is a more distinct difference between problem instances with one domain and those with two domains: for the former STORM-TMO, again the multi-objective solution, performs very well and is always able to find a useful setup. On the other hand, when multiple domains come into play none of the implementations is able to find a useful setup most of the time, which indicates that this approach is not suited for this particular purpose.

Since most problems only occur for larger instances, this could be due to the fact that the current implementation does not learn anything from the correlation between different genes. To investigate whether or not this is the problem, it is interesting to see if the use of *linkage learning* improves the performance of the solver. More on linkage learning in genetic algorithms can be found in Newman [16].

In conclusion, based on the results presented in this thesis the MOGA implementations, STORM-TMM and STORM-TMO, are the most promising. The results of recipes with two domains did show that the population size for precomputation should be kept big, at least higher than 100, and improvement is needed before it can be relied upon to find useful sample setups.

The next step towards implementation of STORM-TMM in the finishing mill is more extensive testing, since the number of problem instances examined here is very limited and a wider variety of instances is needed to substantiate the conclusions. Further, the experiments were conducted with an approximation of the true system and thus it is not guaranteed that the real time solution will produce similar results. Simulation of the system should produce a decisive answer to whether the extension performs well enough. With respect to the problem instances that have multiple domains, it should be examined if the threshold of 20% relative to the reference value might be to high of an aim within in the limited time and whether setups with higher fitness values can be allowed. If not, the choice should be made to look for further improvement of STORM-TMM for samples over the length of the ROT and of STORM-TSM for samples with different objectives or search for another approach to the problem. As mentioned before, improvements could be found in the incorporation of linkage learning or increasing the population size of precomputation.

Lastly, this research did not extend into the examination of the differences between the sample setups found. If the setups were send straight to the controller it is important that switching between two setups requires a minimum of control operations, because these operations are costly and take time. This is a requirement that is not incorporated into the proposed methods, but further research is required on how to make that happen.

A
Choice of Selection Operators

This research also investigates the influence of using different selection methods on the performance of the implementations. Based on [18], four possible operators were chosen: fitness and tournament selection as described in item 3.1.2 and roulette wheel and linear rank selection.

The roulette wheel method is the only method here that does not take the fitness of the individuals into account, but simply selects individuals at random and adds it to the next generation until it holds enough individuals. This technique is included in the experiments because it is very fast, though it can lead to premature convergence to a local optimum.

Finally, linear rank selection is an adaptation of the roulette wheel method. It assigns a selection probability to every individual proportional to its *rank*. The rank that is awarded to an individual depends on its relative fitness, where the best individual gets rank n and the worst individual gets rank 1. The probability of selection is then described as

$$p_i = \frac{rank_i}{n(n-1)}. \tag{A.1}$$

Then selection works as a roulette wheel, though complying to the assigned probabilities. The problem of premature convergence is thereby fixed, however at the expense of the speed advantage.

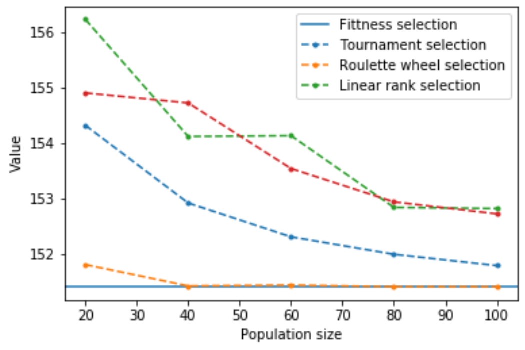
A.1. Methodology

To see which of these four operators is most likely to enhance the performance of the implementations, experiments were performed with STORM-U while changing the selection operator and comparing the performance for different population sizes. The best combination should perform well on two quality measures: proximity to the optimal solution and population diversity. Preferably, the population size is kept as small as possible to guarantee faster computation.

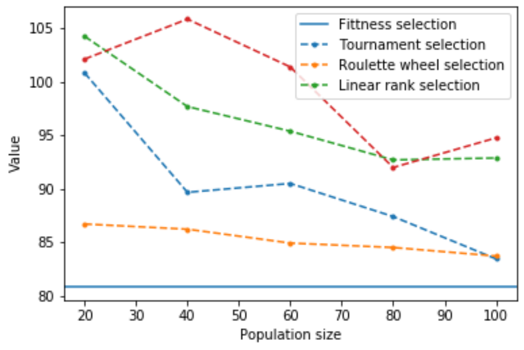
For each of the four selection methods the GA was run for 400 iterations with a population size $n = \{20, 40, 60, 80, 100\}$ for both recipe 1 and recipe 2 as introduced in chapter 4. This process was repeated 50 times, to eliminate the influence of the stochastic nature of the algorithm. Statistical analysis shows that this number of repetitions is enough, since the p-values of each set of measurements are extremely close to zero. See Table A.1 and Table A.2 for recipe 1, the results for recipe 2 are very similar and therefore not attached here.

A.2. Results

From the graphs in Figure A.1 the conclusion can be drawn that fitness selection and tournament selection perform best on both measures, even with a smaller population size. Therefore, the other two selection methods will not be taken into account in further experiments. When taking the diversity into account as plotted in Figure A.2, roulette wheel and linear rank selection have a quite constant diversity rate which can be expected due to the nature of the operators. Fitness and tournament selection remain somewhere just above half of the population to be equal to at least one other individual, which could be improved but will suffice to avoid local premature convergence.

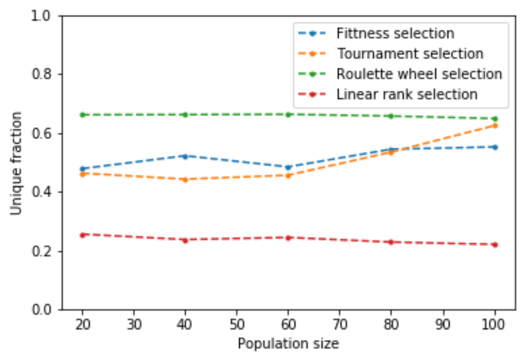


(a) Results for recipe 1

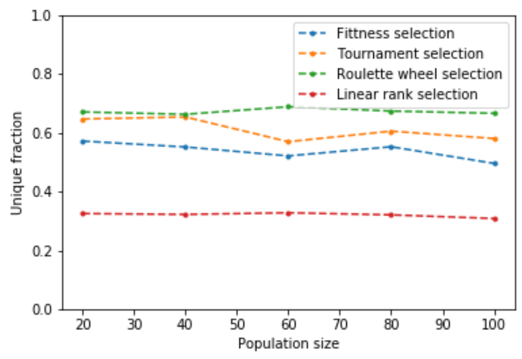


(b) Results for recipe 2

Figure A.1: Performance of different selection methods with respect to the found value. The horizontal line indicates the reference value, which is assumed to be optimal



(a) Results for recipe 1



(b) Results for recipe 2

Figure A.2: Performance of different selection methods with respect to the fraction of unique individuals

Of course, the set of selection methods considered here is limited and many others could be taken into account in future research. Here the choice was made to focus on the most common operators since the aim of this research is to show whether or not an informed start has a positive influence on the performance of the algorithm, though it is very well possible that another set of operators performs better.

Table A.1: Statistical significance for optimal value in population for recipe 1. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%.

Group1	Group2	M1-M2	p
Sort_20	Tournament_20	2.519562	***
Sort_20	Roulette_20	-1.916086	***
Sort_20	LRS_20	-0.585588	***
Sort_20	Sort_40	1.404782	***
Sort_20	Tournament_40	2.904911	***
Sort_20	Roulette_40	0.201638	***
Sort_20	LRS_40	-0.404294	***
Sort_20	Sort_60	2.017503	***
Sort_20	Tournament_60	2.889518	***
Sort_20	Roulette_60	0.187517	***
Sort_20	LRS_60	0.782020	***
Sort_20	Sort_80	2.332961	***
Sort_20	Tournament_80	2.918986	***

Continuation of Table A.1

Group1	Group2	M1-M2	p
Sort_20	Roulette_80	1.484776	***
Sort_20	LRS_80	1.384661	***
Sort_20	Sort_100	2.536961	***
Sort_20	Tournament_100	2.918986	***
Sort_20	Roulette_100	1.505755	***
Sort_20	LRS_100	1.601591	***
Tournament_20	Roulette_20	-4.435648	***
Tournament_20	LRS_20	-3.105150	***
Tournament_20	Sort_40	-1.114780	***
Tournament_20	Tournament_40	0.385349	***
Tournament_20	Roulette_40	-2.317924	***
Tournament_20	LRS_40	-2.923856	***
Tournament_20	Sort_60	-0.502059	***
Tournament_20	Tournament_60	0.369956	***
Tournament_20	Roulette_60	-2.332045	***
Tournament_20	LRS_60	-1.737542	***
Tournament_20	Sort_80	-0.186601	***
Tournament_20	Tournament_80	0.399424	***
Tournament_20	Roulette_80	-1.034786	***
Tournament_20	LRS_80	-1.134901	***
Tournament_20	Sort_100	0.017399	*
Tournament_20	Tournament_100	0.399424	***
Tournament_20	Roulette_100	-1.013807	***
Tournament_20	LRS_100	-0.917971	***
Roulette_20	LRS_20	1.330498	***
Roulette_20	Sort_40	3.320868	***
Roulette_20	Tournament_40	4.820997	***
Roulette_20	Roulette_40	2.117724	***
Roulette_20	LRS_40	1.511792	***
Roulette_20	Sort_60	3.933589	***
Roulette_20	Tournament_60	4.805604	***
Roulette_20	Roulette_60	2.103603	***
Roulette_20	LRS_60	2.698106	***
Roulette_20	Sort_80	4.249047	***
Roulette_20	Tournament_80	4.835072	***
Roulette_20	Roulette_80	3.400862	***
Roulette_20	LRS_80	3.300747	***
Roulette_20	Sort_100	4.453047	***
Roulette_20	Tournament_100	4.835072	***
Roulette_20	Roulette_100	3.421841	***
Roulette_20	LRS_100	3.517677	***
LRS_20	Sort_40	1.990370	***
LRS_20	Tournament_40	3.490499	***
LRS_20	Roulette_40	0.787226	***
LRS_20	LRS_40	0.181294	***
LRS_20	Sort_60	2.603091	***
LRS_20	Tournament_60	3.475106	***
LRS_20	Roulette_60	0.773105	***
LRS_20	LRS_60	1.367608	***
LRS_20	Sort_80	2.918549	***
LRS_20	Tournament_80	3.504574	***
LRS_20	Roulette_80	2.070364	***
LRS_20	LRS_80	1.970249	***

Continuation of Table A.1

Group1	Group2	M1-M2	p
LRS_20	Sort_100	3.122549	***
LRS_20	Tournament_100	3.504574	***
LRS_20	Roulette_100	2.091343	***
LRS_20	LRS_100	2.187179	***
Sort_40	Tournament_40	1.500129	***
Sort_40	Roulette_40	-1.203144	***
Sort_40	LRS_40	-1.809076	***
Sort_40	Sort_60	0.612721	***
Sort_40	Tournament_60	1.484736	***
Sort_40	Roulette_60	-1.217265	***
Sort_40	LRS_60	-0.622762	***
Sort_40	Sort_80	0.928179	***
Sort_40	Tournament_80	1.514204	***
Sort_40	Roulette_80	0.079994	*
Sort_40	LRS_80	-0.020121	*
Sort_40	Sort_100	1.132179	***
Sort_40	Tournament_100	1.514204	***
Sort_40	Roulette_100	0.100973	*
Sort_40	LRS_100	0.196809	***
Tournament_40	Roulette_40	-2.703273	***
Tournament_40	LRS_40	-3.309205	***
Tournament_40	Sort_60	-0.887408	***
Tournament_40	Tournament_60	-0.015393	*
Tournament_40	Roulette_60	-2.717394	***
Tournament_40	LRS_60	-2.122891	***
Tournament_40	Sort_80	-0.571950	***
Tournament_40	Tournament_80	0.014075	*
Tournament_40	Roulette_80	-1.420135	***
Tournament_40	LRS_80	-1.520250	***
Tournament_40	Sort_100	-0.367950	***
Tournament_40	Tournament_100	0.014075	*
Tournament_40	Roulette_100	-1.399156	***
Tournament_40	LRS_100	-1.303320	***
Roulette_40	LRS_40	-0.605932	***
Roulette_40	Sort_60	1.815865	***
Roulette_40	Tournament_60	2.687880	***
Roulette_40	Roulette_60	-0.014121	*
Roulette_40	LRS_60	0.580382	***
Roulette_40	Sort_80	2.131323	***
Roulette_40	Tournament_80	2.717348	***
Roulette_40	Roulette_80	1.283138	***
Roulette_40	LRS_80	1.183023	***
Roulette_40	Sort_100	2.335323	***
Roulette_40	Tournament_100	2.717348	***
Roulette_40	Roulette_100	1.304117	***
Roulette_40	LRS_100	1.399953	***
LRS_40	Sort_60	2.421797	***
LRS_40	Tournament_60	3.293812	***
LRS_40	Roulette_60	0.591811	***
LRS_40	LRS_60	1.186314	***
LRS_40	Sort_80	2.737255	***
LRS_40	Tournament_80	3.323280	***
LRS_40	Roulette_80	1.889070	***

Continuation of Table A.1

Group1	Group2	M1-M2	p
LRS_40	LRS_80	1.788955	***
LRS_40	Sort_100	2.941255	***
LRS_40	Tournament_100	3.323280	***
LRS_40	Roulette_100	1.910049	***
LRS_40	LRS_100	2.005885	***
Sort_60	Tournament_60	0.872015	***
Sort_60	Roulette_60	-1.829986	***
Sort_60	LRS_60	-1.235483	***
Sort_60	Sort_80	0.315458	***
Sort_60	Tournament_80	0.901483	***
Sort_60	Roulette_80	-0.532727	***
Sort_60	LRS_80	-0.632842	***
Sort_60	Sort_100	0.519458	***
Sort_60	Tournament_100	0.901483	***
Sort_60	Roulette_100	-0.511748	***
Sort_60	LRS_100	-0.415912	***
Tournament_60	Roulette_60	-2.702001	***
Tournament_60	LRS_60	-2.107498	***
Tournament_60	Sort_80	-0.556557	***
Tournament_60	Tournament_80	0.029468	*
Tournament_60	Roulette_80	-1.404742	***
Tournament_60	LRS_80	-1.504857	***
Tournament_60	Sort_100	-0.352557	***
Tournament_60	Tournament_100	0.029468	*
Tournament_60	Roulette_100	-1.383763	***
Tournament_60	LRS_100	-1.287927	***
Roulette_60	LRS_60	0.594503	***
Roulette_60	Sort_80	2.145444	***
Roulette_60	Tournament_80	2.731469	***
Roulette_60	Roulette_80	1.297259	***
Roulette_60	LRS_80	1.197144	***
Roulette_60	Sort_100	2.349444	***
Roulette_60	Tournament_100	2.731469	***
Roulette_60	Roulette_100	1.318238	***
Roulette_60	LRS_100	1.414074	***
LRS_60	Sort_80	1.550941	***
LRS_60	Tournament_80	2.136966	***
LRS_60	Roulette_80	0.702756	***
LRS_60	LRS_80	0.602641	***
LRS_60	Sort_100	1.754941	***
LRS_60	Tournament_100	2.136966	***
LRS_60	Roulette_100	0.723735	***
LRS_60	LRS_100	0.819571	***
Sort_80	Tournament_80	0.586025	***
Sort_80	Roulette_80	-0.848185	***
Sort_80	LRS_80	-0.948300	***
Sort_80	Sort_100	0.204000	***
Sort_80	Tournament_100	0.586025	***
Sort_80	Roulette_100	-0.827206	***
Sort_80	LRS_100	-0.731370	***
Tournament_80	Roulette_80	-1.434210	***
Tournament_80	LRS_80	-1.534325	***
Tournament_80	Sort_100	-0.382025	***

Continuation of Table A.1

Group1	Group2	M1-M2	p
Tournament_80	Tournament_100	0.000000	*
Tournament_80	Roulette_100	-1.413231	***
Tournament_80	LRS_100	-1.317395	***
Roulette_80	LRS_80	-0.100115	*
Roulette_80	Sort_100	1.052185	***
Roulette_80	Tournament_100	1.434210	***
Roulette_80	Roulette_100	0.020979	*
Roulette_80	LRS_100	0.116815	*
LRS_80	Sort_100	1.152300	***
LRS_80	Tournament_100	1.534325	***
LRS_80	Roulette_100	0.121094	*
LRS_80	LRS_100	0.216930	***
Sort_100	Tournament_100	0.382025	***
Sort_100	Roulette_100	-1.031206	***
Sort_100	LRS_100	-0.935370	***
Tournament_100	Roulette_100	-1.413231	***
Tournament_100	LRS_100	-1.317395	***
Roulette_100	LRS_100	0.095836	*

Table A.2: Statistical significance for number of unique individuals in elite population for recipe 1. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%.

Group1	Group2	M1-M2	p
Sort_20	Tournament_20	0.015500	**
Sort_20	Roulette_20	-0.183000	***
Sort_20	LRS_20	0.222500	***
Sort_20	Sort_40	-0.043750	***
Sort_20	Tournament_40	0.035750	***
Sort_20	Roulette_40	-0.183500	***
Sort_20	LRS_40	0.240750	***
Sort_20	Sort_60	-0.006000	*
Sort_20	Tournament_60	0.022333	***
Sort_20	Roulette_60	-0.184500	***
Sort_20	LRS_60	0.233667	***
Sort_20	Sort_80	-0.066000	***
Sort_20	Tournament_80	-0.056000	***
Sort_20	Roulette_80	-0.178625	***
Sort_20	LRS_80	0.249375	***
Sort_20	Sort_100	-0.073700	***
Sort_20	Tournament_100	-0.145900	***
Sort_20	Roulette_100	-0.170300	***
Sort_20	LRS_100	0.257200	***
Tournament_20	Roulette_20	-0.198500	***
Tournament_20	LRS_20	0.207000	***
Tournament_20	Sort_40	-0.059250	***
Tournament_20	Tournament_40	0.020250	***
Tournament_20	Roulette_40	-0.199000	***
Tournament_20	LRS_40	0.225250	***
Tournament_20	Sort_60	-0.021500	***
Tournament_20	Tournament_60	0.006833	*
Tournament_20	Roulette_60	-0.200000	***
Tournament_20	LRS_60	0.218167	***

Continuation of Table A.2

Group1	Group2	M1-M2	p
Tournament_20	Sort_80	-0.081500	***
Tournament_20	Tournament_80	-0.071500	***
Tournament_20	Roulette_80	-0.194125	***
Tournament_20	LRS_80	0.233875	***
Tournament_20	Sort_100	-0.089200	***
Tournament_20	Tournament_100	-0.161400	***
Tournament_20	Roulette_100	-0.185800	***
Tournament_20	LRS_100	0.241700	***
Roulette_20	LRS_20	0.405500	***
Roulette_20	Sort_40	0.139250	***
Roulette_20	Tournament_40	0.218750	***
Roulette_20	Roulette_40	-0.000500	*
Roulette_20	LRS_40	0.423750	***
Roulette_20	Sort_60	0.177000	***
Roulette_20	Tournament_60	0.205333	***
Roulette_20	Roulette_60	-0.001500	*
Roulette_20	LRS_60	0.416667	***
Roulette_20	Sort_80	0.117000	***
Roulette_20	Tournament_80	0.127000	***
Roulette_20	Roulette_80	0.004375	*
Roulette_20	LRS_80	0.432375	***
Roulette_20	Sort_100	0.109300	***
Roulette_20	Tournament_100	0.037100	***
Roulette_20	Roulette_100	0.012700	**
Roulette_20	LRS_100	0.440200	***
LRS_20	Sort_40	-0.266250	***
LRS_20	Tournament_40	-0.186750	***
LRS_20	Roulette_40	-0.406000	***
LRS_20	LRS_40	0.018250	***
LRS_20	Sort_60	-0.228500	***
LRS_20	Tournament_60	-0.200167	***
LRS_20	Roulette_60	-0.407000	***
LRS_20	LRS_60	0.011167	*
LRS_20	Sort_80	-0.288500	***
LRS_20	Tournament_80	-0.278500	***
LRS_20	Roulette_80	-0.401125	***
LRS_20	LRS_80	0.026875	***
LRS_20	Sort_100	-0.296200	***
LRS_20	Tournament_100	-0.368400	***
LRS_20	Roulette_100	-0.392800	***
LRS_20	LRS_100	0.034700	***
Sort_40	Tournament_40	0.079500	***
Sort_40	Roulette_40	-0.139750	***
Sort_40	LRS_40	0.284500	***
Sort_40	Sort_60	0.037750	***
Sort_40	Tournament_60	0.066083	***
Sort_40	Roulette_60	-0.140750	***
Sort_40	LRS_60	0.277417	***
Sort_40	Sort_80	-0.022250	***
Sort_40	Tournament_80	-0.012250	**
Sort_40	Roulette_80	-0.134875	***
Sort_40	LRS_80	0.293125	***
Sort_40	Sort_100	-0.029950	***

Continuation of Table A.2

Group1	Group2	M1-M2	p
Sort_40	Tournament_100	-0.102150	***
Sort_40	Roulette_100	-0.126550	***
Sort_40	LRS_100	0.300950	***
Tournament_40	Roulette_40	-0.219250	***
Tournament_40	LRS_40	0.205000	***
Tournament_40	Sort_60	-0.041750	***
Tournament_40	Tournament_60	-0.013417	**
Tournament_40	Roulette_60	-0.220250	***
Tournament_40	LRS_60	0.197917	***
Tournament_40	Sort_80	-0.101750	***
Tournament_40	Tournament_80	-0.091750	***
Tournament_40	Roulette_80	-0.214375	***
Tournament_40	LRS_80	0.213625	***
Tournament_40	Sort_100	-0.109450	***
Tournament_40	Tournament_100	-0.181650	***
Tournament_40	Roulette_100	-0.206050	***
Tournament_40	LRS_100	0.221450	***
Roulette_40	LRS_40	0.424250	***
Roulette_40	Sort_60	0.177500	***
Roulette_40	Tournament_60	0.205833	***
Roulette_40	Roulette_60	-0.001000	*
Roulette_40	LRS_60	0.417167	***
Roulette_40	Sort_80	0.117500	***
Roulette_40	Tournament_80	0.127500	***
Roulette_40	Roulette_80	0.004875	*
Roulette_40	LRS_80	0.432875	***
Roulette_40	Sort_100	0.109800	***
Roulette_40	Tournament_100	0.037600	***
Roulette_40	Roulette_100	0.013200	**
Roulette_40	LRS_100	0.440700	***
LRS_40	Sort_60	-0.246750	***
LRS_40	Tournament_60	-0.218417	***
LRS_40	Roulette_60	-0.425250	***
LRS_40	LRS_60	-0.007083	*
LRS_40	Sort_80	-0.306750	***
LRS_40	Tournament_80	-0.296750	***
LRS_40	Roulette_80	-0.419375	***
LRS_40	LRS_80	0.008625	*
LRS_40	Sort_100	-0.314450	***
LRS_40	Tournament_100	-0.386650	***
LRS_40	Roulette_100	-0.411050	***
LRS_40	LRS_100	0.016450	***
Sort_60	Tournament_60	0.028333	***
Sort_60	Roulette_60	-0.178500	***
Sort_60	LRS_60	0.239667	***
Sort_60	Sort_80	-0.060000	***
Sort_60	Tournament_80	-0.050000	***
Sort_60	Roulette_80	-0.172625	***
Sort_60	LRS_80	0.255375	***
Sort_60	Sort_100	-0.067700	***
Sort_60	Tournament_100	-0.139900	***
Sort_60	Roulette_100	-0.164300	***
Sort_60	LRS_100	0.263200	***

Continuation of Table A.2

Group1	Group2	M1-M2	p
Tournament_60	Roulette_60	-0.206833	***
Tournament_60	LRS_60	0.211333	***
Tournament_60	Sort_80	-0.088333	***
Tournament_60	Tournament_80	-0.078333	***
Tournament_60	Roulette_80	-0.200958	***
Tournament_60	LRS_80	0.227042	***
Tournament_60	Sort_100	-0.096033	***
Tournament_60	Tournament_100	-0.168233	***
Tournament_60	Roulette_100	-0.192633	***
Tournament_60	LRS_100	0.234867	***
Roulette_60	LRS_60	0.418167	***
Roulette_60	Sort_80	0.118500	***
Roulette_60	Tournament_80	0.128500	***
Roulette_60	Roulette_80	0.005875	*
Roulette_60	LRS_80	0.433875	***
Roulette_60	Sort_100	0.110800	***
Roulette_60	Tournament_100	0.038600	***
Roulette_60	Roulette_100	0.014200	**
Roulette_60	LRS_100	0.441700	***
LRS_60	Sort_80	-0.299667	***
LRS_60	Tournament_80	-0.289667	***
LRS_60	Roulette_80	-0.412292	***
LRS_60	LRS_80	0.015708	***
LRS_60	Sort_100	-0.307367	***
LRS_60	Tournament_100	-0.379567	***
LRS_60	Roulette_100	-0.403967	***
LRS_60	LRS_100	0.023533	***
Sort_80	Tournament_80	0.010000	*
Sort_80	Roulette_80	-0.112625	***
Sort_80	LRS_80	0.315375	***
Sort_80	Sort_100	-0.007700	*
Sort_80	Tournament_100	-0.079900	***
Sort_80	Roulette_100	-0.104300	***
Sort_80	LRS_100	0.323200	***
Tournament_80	Roulette_80	-0.122625	***
Tournament_80	LRS_80	0.305375	***
Tournament_80	Sort_100	-0.017700	***
Tournament_80	Tournament_100	-0.089900	***
Tournament_80	Roulette_100	-0.114300	***
Tournament_80	LRS_100	0.313200	***
Roulette_80	LRS_80	0.428000	***
Roulette_80	Sort_100	0.104925	***
Roulette_80	Tournament_100	0.032725	***
Roulette_80	Roulette_100	0.008325	*
Roulette_80	LRS_100	0.435825	***
LRS_80	Sort_100	-0.323075	***
LRS_80	Tournament_100	-0.395275	***
LRS_80	Roulette_100	-0.419675	***
LRS_80	LRS_100	0.007825	*
Sort_100	Tournament_100	-0.072200	***
Sort_100	Roulette_100	-0.096600	***
Sort_100	LRS_100	0.330900	***
Tournament_100	Roulette_100	-0.024400	***

Continuation of Table A.2

Group1	Group2	M1-M2	p
Tournament_100	LRS_100	0.403100	***
Roulette_100	LRS_100	0.427500	***

B

Statistical Analysis

This chapter lists all results from statistical analysis of which parts are also included throughout the thesis, though for clarity and neatness the full tables are only included here.

B.1. Multiple models

Table B.1: Statistical significance of relative values for multi-objective focused experiment, with samples over the length of the ROT for the data set with one domain. The *p*-values are depicted as star ratings, where three stars infers a *p*-value below 5%, two stars 5-10% and one star above 10%. All lines comparing various versions of STORM-TMM are highlighted, as these support statements made in section 5.2.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	0.009437	***
Sort_STORM-U	Sort_STORM-TMM-50	0.020234	***
Sort_STORM-U	Sort_STORM-TMM-100	0.020056	***
Sort_STORM-U	Sort_STORM-TMM-200	0.020126	***
Sort_STORM-U	Tournament_STORM-U	-0.001017	***
Sort_STORM-U	Tournament_STORM-TSM	0.009399	***
Sort_STORM-U	Tournament_STORM-TMM-50	0.019633	***
Sort_STORM-U	Tournament_STORM-TMM-100	0.019547	***
Sort_STORM-U	Tournament_STORM-TMM-200	0.020020	***
Sort_STORM-TSM	Sort_STORM-TMM-50	0.010797	***
Sort_STORM-TSM	Sort_STORM-TMM-100	0.010618	***
Sort_STORM-TSM	Sort_STORM-TMM-200	0.010689	***
Sort_STORM-TSM	Tournament_STORM-U	-0.010454	***
Sort_STORM-TSM	Tournament_STORM-TSM	-0.000038	*
Sort_STORM-TSM	Tournament_STORM-TMM-50	0.010196	***
Sort_STORM-TSM	Tournament_STORM-TMM-100	0.010110	***
Sort_STORM-TSM	Tournament_STORM-TMM-200	0.010583	***
Sort_STORM-TMM-50	Sort_STORM-TMM-100	-0.000178	*
Sort_STORM-TMM-50	Sort_STORM-TMM-200	-0.000108	*
Sort_STORM-TMM-50	Tournament_STORM-U	-0.021251	***
Sort_STORM-TMM-50	Tournament_STORM-TSM	-0.010835	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-50	-0.000601	*
Sort_STORM-TMM-50	Tournament_STORM-TMM-100	-0.000687	**
Sort_STORM-TMM-50	Tournament_STORM-TMM-200	-0.000214	*
Sort_STORM-TMM-100	Sort_STORM-TMM-200	0.000071	*
Sort_STORM-TMM-100	Tournament_STORM-U	-0.021072	***
Sort_STORM-TMM-100	Tournament_STORM-TSM	-0.010656	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-50	-0.000423	*

Continuation of Table B.1

Group1	Group2	M1-M2	p
Sort_STORM-TMM-100	Tournament_STORM-TMM-100	-0.000509	*
Sort_STORM-TMM-100	Tournament_STORM-TMM-200	-0.000035	*
Sort_STORM-TMM-200	Tournament_STORM-U	-0.021143	***
Sort_STORM-TMM-200	Tournament_STORM-TSM	-0.010727	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-50	-0.000493	*
Sort_STORM-TMM-200	Tournament_STORM-TMM-100	-0.000579	*
Sort_STORM-TMM-200	Tournament_STORM-TMM-200	-0.000106	*
Tournament_STORM-U	Tournament_STORM-TSM	0.010416	***
Tournament_STORM-U	Tournament_STORM-TMM-50	0.020650	***
Tournament_STORM-U	Tournament_STORM-TMM-100	0.020564	***
Tournament_STORM-U	Tournament_STORM-TMM-200	0.021037	***
Tournament_STORM-TSM	Tournament_STORM-TMM-50	0.010234	***
Tournament_STORM-TSM	Tournament_STORM-TMM-100	0.010148	***
Tournament_STORM-TSM	Tournament_STORM-TMM-200	0.010621	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-100	-0.000086	*
Tournament_STORM-TMM-50	Tournament_STORM-TMM-200	0.000387	*
Tournament_STORM-TMM-100	Tournament_STORM-TMM-200	0.000473	*

Table B.2: Statistical significance of the number of iterations for multi-objective focused experiment, with samples over the length of the ROT for the data set with one domain. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%. All lines comparing various versions of STORM-TMM are highlighted, as these support statements made in section 5.2.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	1.564815	***
Sort_STORM-U	Sort_STORM-TMM-50	11.740370	***
Sort_STORM-U	Sort_STORM-TMM-100	10.270370	***
Sort_STORM-U	Sort_STORM-TMM-200	11.302593	***
Sort_STORM-U	Tournament_STORM-U	-0.812963	***
Sort_STORM-U	Tournament_STORM-TSM	-0.624074	***
Sort_STORM-U	Tournament_STORM-TMM-50	9.620370	***
Sort_STORM-U	Tournament_STORM-TMM-100	10.775926	***
Sort_STORM-U	Tournament_STORM-TMM-200	10.539259	***
Sort_STORM-TSM	Sort_STORM-TMM-50	10.175556	***
Sort_STORM-TSM	Sort_STORM-TMM-100	8.705556	***
Sort_STORM-TSM	Sort_STORM-TMM-200	9.737778	***
Sort_STORM-TSM	Tournament_STORM-U	-2.377778	***
Sort_STORM-TSM	Tournament_STORM-TSM	-2.188889	***
Sort_STORM-TSM	Tournament_STORM-TMM-50	8.055556	***
Sort_STORM-TSM	Tournament_STORM-TMM-100	9.211111	***
Sort_STORM-TSM	Tournament_STORM-TMM-200	8.974444	***
Sort_STORM-TMM-50	Sort_STORM-TMM-100	-1.470000	***
Sort_STORM-TMM-50	Sort_STORM-TMM-200	-0.437778	**
Sort_STORM-TMM-50	Tournament_STORM-U	-12.553333	***
Sort_STORM-TMM-50	Tournament_STORM-TSM	-12.364444	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-50	-2.120000	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-100	-0.964444	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-200	-1.201111	***
Sort_STORM-TMM-100	Sort_STORM-TMM-200	1.032222	***
Sort_STORM-TMM-100	Tournament_STORM-U	-11.083333	***
Sort_STORM-TMM-100	Tournament_STORM-TSM	-10.894444	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-50	-0.650000	***

Continuation of Table B.2

Group1	Group2	M1-M2	p
Sort_STORM-TMM-100	Tournament_STORM-TMM-100	0.505556	**
Sort_STORM-TMM-100	Tournament_STORM-TMM-200	0.268889	*
Sort_STORM-TMM-200	Tournament_STORM-U	-12.115556	***
Sort_STORM-TMM-200	Tournament_STORM-TSM	-11.926667	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-50	-1.682222	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-100	-0.526667	**
Sort_STORM-TMM-200	Tournament_STORM-TMM-200	-0.763333	***
Tournament_STORM-U	Tournament_STORM-TSM	0.188889	*
Tournament_STORM-U	Tournament_STORM-TMM-50	10.433333	***
Tournament_STORM-U	Tournament_STORM-TMM-100	11.588889	***
Tournament_STORM-U	Tournament_STORM-TMM-200	11.352222	***
Tournament_STORM-TSM	Tournament_STORM-TMM-50	10.244444	***
Tournament_STORM-TSM	Tournament_STORM-TMM-100	11.400000	***
Tournament_STORM-TSM	Tournament_STORM-TMM-200	11.163333	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-100	1.155556	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-200	0.918889	***
Tournament_STORM-TMM-100	Tournament_STORM-TMM-200	-0.236667	*

Table B.3: Statistical significance of relative values for multi-objective focused experiment, with samples over the length of the ROT for the data set with two domains. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%. All lines comparing STORM-TMM-200 with any other implementation are highlighted, since these support the statements made in section 5.2.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	-0.086125	***
Sort_STORM-U	Sort_STORM-TMM-50	-0.229854	***
Sort_STORM-U	Sort_STORM-TMM-100	-0.309233	***
Sort_STORM-U	Sort_STORM-TMM-200	0.276585	***
Sort_STORM-U	Tournament_STORM-U	0.118297	***
Sort_STORM-U	Tournament_STORM-TSM	-0.009716	*
Sort_STORM-U	Tournament_STORM-TMM-50	-0.074596	***
Sort_STORM-U	Tournament_STORM-TMM-100	-0.120261	***
Sort_STORM-U	Tournament_STORM-TMM-200	0.330218	***
Sort_STORM-TSM	Sort_STORM-TMM-50	-0.143729	***
Sort_STORM-TSM	Sort_STORM-TMM-100	-0.223108	***
Sort_STORM-TSM	Sort_STORM-TMM-200	0.362710	***
Sort_STORM-TSM	Tournament_STORM-U	0.204422	***
Sort_STORM-TSM	Tournament_STORM-TSM	0.076409	***
Sort_STORM-TSM	Tournament_STORM-TMM-50	0.011529	*
Sort_STORM-TSM	Tournament_STORM-TMM-100	-0.034136	***
Sort_STORM-TSM	Tournament_STORM-TMM-200	0.416343	***
Sort_STORM-TMM-50	Sort_STORM-TMM-100	-0.079379	***
Sort_STORM-TMM-50	Sort_STORM-TMM-200	0.506439	***
Sort_STORM-TMM-50	Tournament_STORM-U	0.348151	***
Sort_STORM-TMM-50	Tournament_STORM-TSM	0.220138	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-50	0.155258	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-100	0.109592	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-200	0.560071	***
Sort_STORM-TMM-100	Sort_STORM-TMM-200	0.585818	***
Sort_STORM-TMM-100	Tournament_STORM-U	0.427530	***
Sort_STORM-TMM-100	Tournament_STORM-TSM	0.299517	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-50	0.234637	***

Continuation of Table B.3

Group1	Group2	M1-M2	p
Sort_STORM-TMM-100	Tournament_STORM-TMM-100	0.188972	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-200	0.639451	***
Sort_STORM-TMM-200	Tournament_STORM-U	-0.158288	***
Sort_STORM-TMM-200	Tournament_STORM-TSM	-0.286301	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-50	-0.351181	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-100	-0.396847	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-200	0.053632	***
Tournament_STORM-U	Tournament_STORM-TSM	-0.128013	***
Tournament_STORM-U	Tournament_STORM-TMM-50	-0.192894	***
Tournament_STORM-U	Tournament_STORM-TMM-100	-0.238559	***
Tournament_STORM-U	Tournament_STORM-TMM-200	0.211920	***
Tournament_STORM-TSM	Tournament_STORM-TMM-50	-0.064880	***
Tournament_STORM-TSM	Tournament_STORM-TMM-100	-0.110545	***
Tournament_STORM-TSM	Tournament_STORM-TMM-200	0.339934	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-100	-0.045665	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-200	0.404814	***
Tournament_STORM-TMM-100	Tournament_STORM-TMM-200	0.450479	***

Table B.4: Statistical significance of the number of iterations for multi-objective focused experiment, with samples over the length of the ROT for the data set with two domains. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%. All lines comparing STORM-TMM-200 with any other implementation are highlighted, since these support the statements made in section 5.2.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	4.068695	***
Sort_STORM-U	Sort_STORM-TMM-50	-18.468607	***
Sort_STORM-U	Sort_STORM-TMM-100	-19.444797	***
Sort_STORM-U	Sort_STORM-TMM-200	6.479012	***
Sort_STORM-U	Tournament_STORM-U	-27.016049	***
Sort_STORM-U	Tournament_STORM-TSM	-17.100353	***
Sort_STORM-U	Tournament_STORM-TMM-50	-47.449559	***
Sort_STORM-U	Tournament_STORM-TMM-100	-44.760670	***
Sort_STORM-U	Tournament_STORM-TMM-200	-14.173765	***
Sort_STORM-TSM	Sort_STORM-TMM-50	-22.537302	***
Sort_STORM-TSM	Sort_STORM-TMM-100	-23.513492	***
Sort_STORM-TSM	Sort_STORM-TMM-200	2.410317	***
Sort_STORM-TSM	Tournament_STORM-U	-31.084744	***
Sort_STORM-TSM	Tournament_STORM-TSM	-21.169048	***
Sort_STORM-TSM	Tournament_STORM-TMM-50	-51.518254	***
Sort_STORM-TSM	Tournament_STORM-TMM-100	-48.829365	***
Sort_STORM-TSM	Tournament_STORM-TMM-200	-18.242460	***
Sort_STORM-TMM-50	Sort_STORM-TMM-100	-0.976190	*
Sort_STORM-TMM-50	Sort_STORM-TMM-200	24.947619	***
Sort_STORM-TMM-50	Tournament_STORM-U	-8.547443	***
Sort_STORM-TMM-50	Tournament_STORM-TSM	1.368254	**
Sort_STORM-TMM-50	Tournament_STORM-TMM-50	-28.980952	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-100	-26.292063	***
Sort_STORM-TMM-50	Tournament_STORM-TMM-200	4.294841	***
Sort_STORM-TMM-100	Sort_STORM-TMM-200	25.923810	***
Sort_STORM-TMM-100	Tournament_STORM-U	-7.571252	***
Sort_STORM-TMM-100	Tournament_STORM-TSM	2.344444	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-50	-28.004762	***

Continuation of Table B.4

Group1	Group2	M1-M2	p
Sort_STORM-TMM-100	Tournament_STORM-TMM-100	-25.315873	***
Sort_STORM-TMM-100	Tournament_STORM-TMM-200	5.271032	***
Sort_STORM-TMM-200	Tournament_STORM-U	-33.495062	***
Sort_STORM-TMM-200	Tournament_STORM-TSM	-23.579365	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-50	-53.928571	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-100	-51.239683	***
Sort_STORM-TMM-200	Tournament_STORM-TMM-200	-20.652778	***
Tournament_STORM-U	Tournament_STORM-TSM	9.915697	***
Tournament_STORM-U	Tournament_STORM-TMM-50	-20.433510	***
Tournament_STORM-U	Tournament_STORM-TMM-100	-17.744621	***
Tournament_STORM-U	Tournament_STORM-TMM-200	12.842284	***
Tournament_STORM-TSM	Tournament_STORM-TMM-50	-30.349206	***
Tournament_STORM-TSM	Tournament_STORM-TMM-100	-27.660317	***
Tournament_STORM-TSM	Tournament_STORM-TMM-200	2.926587	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-100	2.688889	***
Tournament_STORM-TMM-50	Tournament_STORM-TMM-200	33.275794	***
Tournament_STORM-TMM-100	Tournament_STORM-TMM-200	30.586905	***

B.2. Multiple Objectives

Table B.5: Statistical significance of relative values for multi-objective focused experiment, with samples having different objectives for the data set with one domain. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%. All lines comparing various versions of STORM-TMO are highlighted, as these support statements made in section 5.3.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	-0.008908	***
Sort_STORM-U	Sort_STORM-TMO-50	0.017369	***
Sort_STORM-U	Sort_STORM-TMO-100	0.019305	***
Sort_STORM-U	Sort_STORM-TMO-200	0.019332	***
Sort_STORM-U	Tournament_STORM-U	-0.001262	***
Sort_STORM-U	Tournament_STORM-TSM	-0.001238	***
Sort_STORM-U	Tournament_STORM-TMO-50	0.017845	***
Sort_STORM-U	Tournament_STORM-TMO-100	0.019076	***
Sort_STORM-U	Tournament_STORM-TMO-200	0.018525	***
Sort_STORM-TSM	Sort_STORM-TMO-50	0.026277	***
Sort_STORM-TSM	Sort_STORM-TMO-100	0.028213	***
Sort_STORM-TSM	Sort_STORM-TMO-200	0.028240	***
Sort_STORM-TSM	Tournament_STORM-U	0.007646	***
Sort_STORM-TSM	Tournament_STORM-TSM	0.007670	***
Sort_STORM-TSM	Tournament_STORM-TMO-50	0.026753	***
Sort_STORM-TSM	Tournament_STORM-TMO-100	0.027984	***
Sort_STORM-TSM	Tournament_STORM-TMO-200	0.027433	***
Sort_STORM-TMO-50	Sort_STORM-TMO-100	0.001935	***
Sort_STORM-TMO-50	Sort_STORM-TMO-200	0.001963	***
Sort_STORM-TMO-50	Tournament_STORM-U	-0.018631	***
Sort_STORM-TMO-50	Tournament_STORM-TSM	-0.018607	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-50	0.000475	*
Sort_STORM-TMO-50	Tournament_STORM-TMO-100	0.001706	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-200	0.001156	***
Sort_STORM-TMO-100	Sort_STORM-TMO-200	0.000027	*
Sort_STORM-TMO-100	Tournament_STORM-U	-0.020567	***

Continuation of Table B.5

Group1	Group2	M1-M2	p
Sort_STORM-TMO-100	Tournament_STORM-TSM	-0.020543	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-50	-0.001460	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-100	-0.000229	*
Sort_STORM-TMO-100	Tournament_STORM-TMO-200	-0.000779	**
Sort_STORM-TMO-200	Tournament_STORM-U	-0.020594	***
Sort_STORM-TMO-200	Tournament_STORM-TSM	-0.020570	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-50	-0.001487	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-100	-0.000256	*
Sort_STORM-TMO-200	Tournament_STORM-TMO-200	-0.000807	**
Tournament_STORM-U	Tournament_STORM-TSM	0.000024	*
Tournament_STORM-U	Tournament_STORM-TMO-50	0.019106	***
Tournament_STORM-U	Tournament_STORM-TMO-100	0.020337	***
Tournament_STORM-U	Tournament_STORM-TMO-200	0.019787	***
Tournament_STORM-TSM	Tournament_STORM-TMO-50	0.019082	***
Tournament_STORM-TSM	Tournament_STORM-TMO-100	0.020313	***
Tournament_STORM-TSM	Tournament_STORM-TMO-200	0.019763	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-100	0.001231	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-200	0.000681	**
Tournament_STORM-TMO-100	Tournament_STORM-TMO-200	-0.000550	*

Table B.6: Statistical significance of the number of iterations for multi-objective focused experiment, with samples having different sets of objectives for the data set with one domain. The *p*-values are depicted as star ratings, where three stars infers a *p*-value below 5%, two stars 5-10% and one star above 10%. All lines comparing various versions of STORM-TMO are highlighted, as these support statements made in section 5.3.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	5.101587	***
Sort_STORM-U	Sort_STORM-TMO-50	-5.971032	***
Sort_STORM-U	Sort_STORM-TMO-100	-3.100794	***
Sort_STORM-U	Sort_STORM-TMO-200	-1.960317	***
Sort_STORM-U	Tournament_STORM-U	-2.128175	***
Sort_STORM-U	Tournament_STORM-TSM	2.545635	***
Sort_STORM-U	Tournament_STORM-TMO-50	-6.775794	***
Sort_STORM-U	Tournament_STORM-TMO-100	-2.860317	***
Sort_STORM-U	Tournament_STORM-TMO-200	-2.593651	***
Sort_STORM-TSM	Sort_STORM-TMO-50	-11.072619	***
Sort_STORM-TSM	Sort_STORM-TMO-100	-8.202381	***
Sort_STORM-TSM	Sort_STORM-TMO-200	-7.061905	***
Sort_STORM-TSM	Tournament_STORM-U	-7.229762	***
Sort_STORM-TSM	Tournament_STORM-TSM	-2.555952	***
Sort_STORM-TSM	Tournament_STORM-TMO-50	-11.877381	***
Sort_STORM-TSM	Tournament_STORM-TMO-100	-7.961905	***
Sort_STORM-TSM	Tournament_STORM-TMO-200	-7.695238	***
Sort_STORM-TMO-50	Sort_STORM-TMO-100	2.870238	***
Sort_STORM-TMO-50	Sort_STORM-TMO-200	4.010714	***
Sort_STORM-TMO-50	Tournament_STORM-U	3.842857	***
Sort_STORM-TMO-50	Tournament_STORM-TSM	8.516667	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-50	-0.804762	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-100	3.110714	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-200	3.377381	***
Sort_STORM-TMO-100	Sort_STORM-TMO-200	1.140476	***
Sort_STORM-TMO-100	Tournament_STORM-U	0.972619	***

Continuation of Table B.6

Group1	Group2	M1-M2	p
Sort_STORM-TMO-100	Tournament_STORM-TSM	5.646429	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-50	-3.675000	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-100	0.240476	*
Sort_STORM-TMO-100	Tournament_STORM-TMO-200	0.507143	***
Sort_STORM-TMO-200	Tournament_STORM-U	-0.167857	*
Sort_STORM-TMO-200	Tournament_STORM-TSM	4.505952	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-50	-4.815476	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-100	-0.900000	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-200	-0.633333	***
Tournament_STORM-U	Tournament_STORM-TSM	4.673810	***
Tournament_STORM-U	Tournament_STORM-TMO-50	-4.647619	***
Tournament_STORM-U	Tournament_STORM-TMO-100	-0.732143	***
Tournament_STORM-U	Tournament_STORM-TMO-200	-0.465476	**
Tournament_STORM-TSM	Tournament_STORM-TMO-50	-9.321429	***
Tournament_STORM-TSM	Tournament_STORM-TMO-100	-5.405952	***
Tournament_STORM-TSM	Tournament_STORM-TMO-200	-5.139286	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-100	3.915476	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-200	4.182143	***
Tournament_STORM-TMO-100	Tournament_STORM-TMO-200	0.266667	*

Table B.7: Statistical significance of relative values for multi-objective focused experiment, with samples having different objectives for the data set with two domains. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%. All rows comparing STORM-TSM to any of the other implementations are highlighted, as these support statements in section 5.3.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	0.049258	***
Sort_STORM-U	Sort_STORM-TMO-50	-0.025940	***
Sort_STORM-U	Sort_STORM-TMO-100	-0.017783	***
Sort_STORM-U	Sort_STORM-TMO-200	0.014780	***
Sort_STORM-U	Tournament_STORM-U	0.028339	***
Sort_STORM-U	Tournament_STORM-TSM	0.057785	***
Sort_STORM-U	Tournament_STORM-TMO-50	0.022255	***
Sort_STORM-U	Tournament_STORM-TMO-100	0.020125	***
Sort_STORM-U	Tournament_STORM-TMO-200	0.048728	***
Sort_STORM-TSM	Sort_STORM-TMO-50	-0.075197	***
Sort_STORM-TSM	Sort_STORM-TMO-100	-0.067041	***
Sort_STORM-TSM	Sort_STORM-TMO-200	-0.034477	***
Sort_STORM-TSM	Tournament_STORM-U	-0.020919	***
Sort_STORM-TSM	Tournament_STORM-TSM	0.008528	***
Sort_STORM-TSM	Tournament_STORM-TMO-50	-0.027002	***
Sort_STORM-TSM	Tournament_STORM-TMO-100	-0.029133	***
Sort_STORM-TSM	Tournament_STORM-TMO-200	-0.000529	*
Sort_STORM-TMO-50	Sort_STORM-TMO-100	0.008156	***
Sort_STORM-TMO-50	Sort_STORM-TMO-200	0.040720	***
Sort_STORM-TMO-50	Tournament_STORM-U	0.054278	***
Sort_STORM-TMO-50	Tournament_STORM-TSM	0.083725	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-50	0.048195	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-100	0.046064	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-200	0.074668	***
Sort_STORM-TMO-100	Sort_STORM-TMO-200	0.032563	***
Sort_STORM-TMO-100	Tournament_STORM-U	0.046122	***

Continuation of Table B.7

Group1	Group2	M1-M2	p
Sort_STORM-TMO-100	Tournament_STORM-TSM	0.075568	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-50	0.040038	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-100	0.037908	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-200	0.066511	***
Sort_STORM-TMO-200	Tournament_STORM-U	0.013558	***
Sort_STORM-TMO-200	Tournament_STORM-TSM	0.043005	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-50	0.007475	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-100	0.005345	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-200	0.033948	***
Tournament_STORM-U	Tournament_STORM-TSM	0.029447	***
Tournament_STORM-U	Tournament_STORM-TMO-50	-0.006083	***
Tournament_STORM-U	Tournament_STORM-TMO-100	-0.008214	***
Tournament_STORM-U	Tournament_STORM-TMO-200	0.020390	***
Tournament_STORM-TSM	Tournament_STORM-TMO-50	-0.035530	***
Tournament_STORM-TSM	Tournament_STORM-TMO-100	-0.037660	***
Tournament_STORM-TSM	Tournament_STORM-TMO-200	-0.009057	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-100	-0.002130	*
Tournament_STORM-TMO-50	Tournament_STORM-TMO-200	0.026473	***
Tournament_STORM-TMO-100	Tournament_STORM-TMO-200	0.028604	***

Table B.8: Statistical significance of the number of iterations for multi-objective focused experiment, with samples having different sets of objectives for the data set with two domains. The p-values are depicted as star ratings, where three stars infers a p-value below 5%, two stars 5-10% and one star above 10%.

Group1	Group2	M1-M2	p
Sort_STORM-U	Sort_STORM-TSM	23.097650	***
Sort_STORM-U	Sort_STORM-TMO-50	-0.261710	*
Sort_STORM-U	Sort_STORM-TMO-100	4.048543	***
Sort_STORM-U	Sort_STORM-TMO-200	6.106876	***
Sort_STORM-U	Tournament_STORM-U	-35.540405	***
Sort_STORM-U	Tournament_STORM-TSM	8.477412	***
Sort_STORM-U	Tournament_STORM-TMO-50	-40.968891	***
Sort_STORM-U	Tournament_STORM-TMO-100	-43.420207	***
Sort_STORM-U	Tournament_STORM-TMO-200	-37.388957	***
Sort_STORM-TSM	Sort_STORM-TMO-50	-23.359360	***
Sort_STORM-TSM	Sort_STORM-TMO-100	-19.049107	***
Sort_STORM-TSM	Sort_STORM-TMO-200	-16.990774	***
Sort_STORM-TSM	Tournament_STORM-U	-58.638055	***
Sort_STORM-TSM	Tournament_STORM-TSM	-14.620238	***
Sort_STORM-TSM	Tournament_STORM-TMO-50	-64.066541	***
Sort_STORM-TSM	Tournament_STORM-TMO-100	-66.517857	***
Sort_STORM-TSM	Tournament_STORM-TMO-200	-60.486607	***
Sort_STORM-TMO-50	Sort_STORM-TMO-100	4.310253	***
Sort_STORM-TMO-50	Sort_STORM-TMO-200	6.368586	***
Sort_STORM-TMO-50	Tournament_STORM-U	-35.278695	***
Sort_STORM-TMO-50	Tournament_STORM-TSM	8.739122	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-50	-40.707182	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-100	-43.158497	***
Sort_STORM-TMO-50	Tournament_STORM-TMO-200	-37.127247	***
Sort_STORM-TMO-100	Sort_STORM-TMO-200	2.058333	***
Sort_STORM-TMO-100	Tournament_STORM-U	-39.588948	***
Sort_STORM-TMO-100	Tournament_STORM-TSM	4.428869	***

Continuation of Table B.8

Group1	Group2	M1-M2	p
Sort_STORM-TMO-100	Tournament_STORM-TMO-50	-45.017434	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-100	-47.468750	***
Sort_STORM-TMO-100	Tournament_STORM-TMO-200	-41.437500	***
Sort_STORM-TMO-200	Tournament_STORM-U	-41.647281	***
Sort_STORM-TMO-200	Tournament_STORM-TSM	2.370536	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-50	-47.075768	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-100	-49.527083	***
Sort_STORM-TMO-200	Tournament_STORM-TMO-200	-43.495833	***
Tournament_STORM-U	Tournament_STORM-TSM	44.017817	***
Tournament_STORM-U	Tournament_STORM-TMO-50	-5.428486	***
Tournament_STORM-U	Tournament_STORM-TMO-100	-7.879802	***
Tournament_STORM-U	Tournament_STORM-TMO-200	-1.848552	***
Tournament_STORM-TSM	Tournament_STORM-TMO-50	-49.446303	***
Tournament_STORM-TSM	Tournament_STORM-TMO-100	-51.897619	***
Tournament_STORM-TSM	Tournament_STORM-TMO-200	-45.866369	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-100	-2.451316	***
Tournament_STORM-TMO-50	Tournament_STORM-TMO-200	3.579934	***
Tournament_STORM-TMO-100	Tournament_STORM-TMO-200	6.031250	***

Bibliography

[1] Zahra Alizadeh Afrouzy, Seyed Hadi Nasseri, and Iraj Mahdavi. A genetic algorithm for supply chain configuration with new product development. *Computers & Industrial Engineering*, 101:440–454, 2016.

[2] Dipankar Dasgupta, German Hernandez, Deon Garrett, Kalyan Vejandla, Aishwarya Kaushal, Ramjee Yerneni, and James Simien. A Comparison Of Multiobjective Evolutionary Algorithms with Informed Initialization and Kuhn-Munkres Algorithm For The Sailor Assignment Problem. pages 2129–2134. Proceedings of the 10th annual conference companion on Genetic and evolutionary computation, 2008.

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[4] Konstantinos P. Ferentinos and Theodore A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031–1051, 2007.

[5] Banda Gerald. A brief review of independent, dependent and one sample t-test. *International Journal of Applied Mathematics and Theoretical Physics*, 4(2):50–54, 2018.

[6] Abdelhalim Hiassat, Ali Diabat, and Iyad Rahwan. A genetic algorithm approach for location-inventory-routing problem with perishable products. *Journal of manufacturing systems*, 42:93–103, 2017.

[7] John Holland. Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*, 1975.

[8] Kelli D. Humbird, J. Luc Peterson, and Ryan G. Mcclarren. Deep Neural Network Initialization With Decision Trees. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1286–1295, 2019. ISSN 21622388.

[9] Jungtaek Kim, Saehoon Kim, and Seungjin Choi. Learning to warm-start bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*, 2017.

[10] Shyam P. Kodali, Rajesh Kudikala, and K. Deb. Multi-objective optimization of surface grinding process using NSGA II. *Proceedings - 1st International Conference on Emerging Trends in Engineering and Technology*, pages 763–767, 2008.

[11] Slawomir Koziel and Xin-She Yang. *Computational optimization, methods and algorithms*, volume 356. Springer, 2011.

[12] Edward H. Livingston. Who was student and why do we care so much about his t-test? *Journal of Surgical Research*, 118(1):58–65, 2004.

[13] Tobia Marcucci and Russ Tedrake. Warm Start of Mixed-Integer Programs for Model Predictive Control of Hybrid Systems. *IEEE Transactions on Automatic Control*, 9286(c):1–1, 2020.

[14] Morteza Montazeri-Gh, Amir Poursamad, and Babak Ghalichi. Application of genetic algorithm for optimization of control strategy in parallel hybrid electric vehicles. *Metal Finishing*, 104(6):420–435, 2006.

[15] Seiichiro Morizawa, Taku Nonomura, Shigeru Obayashi, Akira Oyama, and Kozo Fujii. Multiobjective design exploration of propeller airfoils at low-reynolds and high-mach number conditions towards mars airplane. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 14(30):47–53, 2016.

[16] David R Newman. The use of linkage learning in genetic algorithms, 2006.

[17] Lorenzo Perino, Akihiro Fujii, Tsuyoshi Waku, Akira Kobayashi, Satoru Hiwa, and Tomoyuki Hiroyasu. Solution exploration using multi-objective genetic algorithm for determining experiment candidate. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1584–1589, 2018.

[18] Nisha Saini. Review of selection methods in genetic algorithms. *International Journal of Engineering and Computer Science*, 6(12):22261–22263, 2017.

[19] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[20] Elise Whitley and Jonathan Ball. Statistics review 5: Comparison of means. *Critical Care*, 6(5):424, 2002.